VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Informační systém pro evidenci a účtování výpočetního času v prostředí HPC

Information System for HPC Computing Resources Accounting

2018 Jan Krupa

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jan Krupa**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Informační systém pro evidenci a účtování výpočetního času v prostředí HPC
Information System for HPC Computing Resources Accounting

Jazyk vypracování: čeština

Zásady pro vypracování:

Národní superpočítačové centrum IT4Innovations provozuje několik superpočítačů. Tyto jsou zpřístupněny široké škále uživatelů. Uživatelé mohou žádat o projekty, kterým jsou poté přiděleny následně tzv. jádrohodiny. Ty slouží k omezení přístupu k výpočetním zdrojům. V současnosti je tato agenda neúplně podchycena v informačním systému, který nelze již dále rozvíjet. Cílem této práce je navrhnout nové řešení, implementovat jej a pokusit se nahradit stávající systém.

Úkoly:
1. Seznámit se se stávajícím řešením informačního systému.
2. Navrhnout modulární řešení IS formou agenda=modul.
3. Vytvořit systém sběru dat pro evidenci výpočetního času napříč clustery.
4. Vytvořit vhodnou prezentační vrstvu implementující mimo jiné validaci vstupních dat, zabezpečení a role.
5. Srovnat původní a novou verzi systému.

Seznam doporučené odborné literatury:

[1] SMITH, Elliot a Rob NICHOLS. Ruby on Rails Enterprise application development: plan, program, extend ; building a complete Ruby on Rails business application from start to finish. Birmingham [u.a.]: Packt Publ, 2007. ISBN 9781847190857.
[2] AVISON, D. AND FITZGERALD, G. Information Systems Development: Methodologies, Techniques and Tools. McGraw-Hill Higher Education, 2006. ISBN 0077114175.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. David Hrbáč**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.	prof. Ing. Pavel Brandštetter, CSc.
vedoucí katedry	*děkan fakulty*

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 20. dubna 2018 .....

Rád bych poděkoval Ing. Davidu Hrbáčovi za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování diplomové práce.

Abstrakt

Tato diplomová práce je zaměřena na sběr dat spuštěných úloh na výpočetních uzlech národního superpočítačového centra IT4Innovations a návrhu informačního systému pro jejich zobrazení. Nově vytvořený informační systém má nahradit stávající, který nelze nadále rozvíjet a udržovat. je potřeba nahradit všechny stávající procesy, automatizovat opakované činnosti a zjednodušit správu přidělování přístupových práv k výpočetním zdrojům administrátorskému týmu HPC.

Práce je koncipována do dvou logických celků, kdy v první části popisujeme stávající informační systém, jeho výhody i nevýhody a nově navržený informační systém s popisem vývoje a jeho nasazením do běžného provozu. Druhá část je zaměřena na evidenci výpočetního času superpočítačového centra od zadání úlohy až po její uložení do databáze.

Klíčová slova: Informační systém, superpočítačové centrum, výpočetní čas

Abstract

This master thesis focuses on data collecting of running jobs on computing nodes at National Supercomputing Center IT4Innovations in the Czech Republic. Collected data has to be presented by a new information system which should be delivered as a result of this thesis.

This information system is supposed to replace the existing one, which can not be extended and maintained anymore. It is necessary to replace all existing processes, automate repetitive activities, and simplify administration of computing resources for the HPC Admin Team to decrease the workload.

The thesis consists of two logical parts. The first part starts by describing the current information system and explaining advantages and disadvantages of the current solution. The development and deployment of the new system and approach is described then. The second part focuses on data collecting of running jobs at supercomputers. I have described overall process of job lifetime through and through, from the very beginning - user job submission, to the very end - job storing at central database.

Key Words: Information system, supercomputing center, computing time

Obsah

Seznam použitých zkratek a symbolů **13**

Seznam obrázků **15**

Seznam tabulek **17**

Seznam výpisů zdrojového kódu **19**

1 Úvod **21**

2 Informační systém pro správu superpočítačového centra **23**
2.1 Původní informační systém SCCS 23
2.2 Nový informační systém SCS 30
2.3 Nasazení SCS IS do běžného provozu 43

3 Správa evidence výpočetního času **45**
3.1 Přidělení výpočetních zdrojů 45
3.2 Veličina výpočetních zdrojů 45
3.3 Plánovač HPC úloh 46
3.4 Zadání úlohy do HPC systému 46
3.5 Fronty úloh 47
3.6 Kontrola předání úlohy plánovači 49
3.7 Evidence výpočetního času 51
3.8 Výstupní soubory plánovače PBS 51
3.9 Typy záznamů 52
3.10 Zpracování účtovacích souborů 53
3.11 Zpracování záznamů databázovým serverem 57
3.12 Měsíční souhrny využitého výpočetního času 60
3.13 Souhrn správy evidence výpočetního času 61

4 Závěr **63**

Literatura **65**

Přílohy **67**

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
AR	–	Active Record
Apache	–	Apache HTTP Server
BDD	–	Behaviour Driven Development
BLOB	–	Binary Large Object
CD	–	Continuous Delivery
CH	–	Core-Hours
CI	–	Continuous Integration
CMS	–	Content Management System
CPAN	–	Comprehensive Perl Archive Network
DB	–	Databáze
DRY	–	Don't Repeat Yourself
FDD	–	Feature Driven Development
FTP	–	File Transfer Protocol
HPC	–	High-Performance Computing
HTML	–	Hyper Text Markup Language
HTTPS	–	Hypertext Transfer Protocol Secure
HTTP	–	Hypertext Transfer Protocol
IMAP	–	Internet Message Access Protocol
IS	–	Informační Systém
JSON	–	JavaScript Object Notation
LDAP	–	Lightweight Directory Access Protocol
MOTD	–	Message Of The Day
MVC	–	Model View Controller
NCH	–	Normalized Core-Hours
NGINX	–	Engine X web server
ORM	–	Object-relational Mapping
OpenGL	–	Open Graphics Library
PBS	–	Portable Batch System
PI	–	Primary Investigator
POP3	–	Post Office Protocol
RDBMS	–	Relational Database Management System
REST	–	Representational State Transfer
RPC	–	Remote Procedure Call
RSS	–	Rich Site Summary
Rails	–	Ruby On Rails framework

SCCS	–	Super Computing Control System
SCS-API	–	Super Computing Services API
scs-ror	–	Super Computing Services information system on Rails
SCS	–	Super Computing Services
SQL	–	Structured Query Language
TAL	–	Template Attribute Language
TDD	–	Test Driven Development
UML	–	Unified Modeling Language
URL	–	Uniform Resource Locator
VI	–	Virtual Infrastructure
VPN	–	Virtual Private Network
WCH	–	Wall-Clock Core-Hours
WebDav	–	Web-based Distributed Authoring and Versioning
XML	–	eXtensible Markup Language
XP	–	eXtreme Programming
ZMI	–	Zope Interface Management
ZODB	–	Zope Object Database
ZSQL	–	Zope Structured Query Language
Zope	–	Z object publishing environment

Seznam obrázků

1 Základní komponenty Zope aplikačního serveru [1] 24
2 Zpracování příchozího požadavku frameworkem Rails [9] 32
3 GitLab kontinuální integrace 38
4 Vývojový diagram metody *check-access* 50
5 Databázové tabulky pro úlohy a rezervace *check-access* 54
6 DB schéma pro agregaci výpočetního času 57
7 Zjednodušený průběh zpracování úlohy/rezervace účtovacím systémem 62

Seznam tabulek

1 Faktorizace normalizovaných jádrohodin 46
2 Přehled bezprojektových front 48
3 Parametry metody *check-access* 49
4 Popis DB schématu pro agregaci výpočetního času 58
5 Tabulka faktorů výpočetního času superpočítačů Anselm a Salomon 58
6 Souhrn využitého výpočetního času projekty 60

Seznam výpisů zdrojového kódu

1 Zadání úlohy plánovači PBS 46
2 Volání metody check-access 49
3 Návratová hodnota metody check-access 49
4 Účtovací soubor plánovače PBS Pro 51
5 Vláknové zpracování účtovacích souborů 55

1 Úvod

V diplomové práci jsem se soustředil na vývoj informačního systému s evidencí účtování výpočetního času v prostředí HPC. V superpočítačovém centru IT4Innovations přicházíme do styku s různými informacemi, které je potřeba vhodně prezentovat, jednoduše s nimy manipulovat a kontrolovat jejich správnost. Informacemi v tomto prostředí jsou uživatelé, projekty a jejich práva pro využívání výpočetních zdrojů, spuštěné HPC úlohy či zprávy pro notifikaci uživatelů. IS tyto informace eviduje, zpracovává, uchovává, kontroluje, agreguje a poskytuje.

Nově navržený informační systém, má nahradit stávající řešení, které není možné nadále rozvíjet. je potřeba převést všechny definované procesy, které nejsou úplně evidovány stávajícím systémem, zjednodušit jejich správu a pokusit se automatizovat ručně prováděné rutinní úkony. Dále je potřeba rozšířit IS o nové funkcionality, jako je ukládání historie prováděných uživatelských akcí, filtrování záznamů, tvorbu vlastních přehledů, spuštění opakovaných operací podle stanoveného časového plánu, provádění kontrol dat nebo automatické notifikování uživatelů. Informační systém slouží primárně pro administrátorský tým HPC, s jehož pomocí může spravovat přístupová práva uživatelů, vytvářet projekty s přidělením alokace možného využití výpočetních zdrojů nebo plánovat odstávky HPC systému. Systém má být navržen modulárním řešením, kdy jedna agenda systému odpovídá jednomu modulu v kódu programu. Pro splnění tohoto konceptu je nový IS postaven na architektuře MVC.

Práce nadále popisuje průběh zpracování úloh uživatelů spouštěných v prostředí HPC superpočítačového centra IT4Innovations. V této části si popíšeme, jak je možné získat výpočetní čas, jakým způsobem je účtován, jaká omezení jsou kladena pro zadání úlohy na výpočetní uzly, a jak jsou následně úlohy zpracovávány. Zpracování sestává z několika procesů, jejichž úkolem je určit spotřebu výpočetních zdrojů uživatele superpočítače. Informace získané ze spuštěných úloh, spolu se zpracovanými souhrny, jsou prezentovány nově navrženým informačním systémem. Tato evidence nám poskytuje kontrolu nad spotřebou výpočetních zdrojů, přehled celkové využitelnosti superpočítačů, predicki požadovaných zdrojů a možnost diagnostiky uživatelských úloh.

2 Informační systém pro správu superpočítačového centra

Informační systém superpočítačového centra je určen k přidělování a správě výpočetních prostředků pro potřeby administrátorského SCS týmu. V IS jsou vedeny projekty s přidělenou alokací výpočetního času, uživatelé, jejich skupiny, role a diskové kvóty, fronty pro zařazení úloh a výzvy pro podávání projektových žádostí. Eviduje se v něm reálné využití přidělovaného výpočetního času projektům, jehož detaily se dále analyzují. Pomocí IS je administrátorský tým SCS schopen notifikovat uživatele založením tzv. zprávy dne označované jako MOTD nebo publikovat informace ohledně plánovaných výpadků superpočítače agendou *dedicated time*, které využívají další systémy. IS má nastaveny různé kontroly, které jsou schopny upozornit správce o nestandardním chování systému. V současnosti se jedná pouze o interní portál pro potřeby administrátorského týmu SCS, ale do budoucna je plánován rozvoj systému o další agendy a jeho zpřístupnění pro běžné uživatele využívající výpočetní zdroje. Uživatelé by byli schopni přes IS zobrazit přehled o svých projektech, právech a mohli by skrze IS podávat různé požadavky, jako je prodloužení projektu či žádost o navýšení diskové kvóty.

IS správy superpočítačového centra je provázán s dalším IS nazývaným Extranet, který slouží jako tzv. *self-service* portál. Extranet je zaměřen především na zakládání projektových žádostí a jejich zpracování alokační komisí, která rozhoduje o tom, jestli budou k žádostem přiděleny výpočetní zdroje. Tento portál také není nadále možno rozvíjet a v současnosti se již pracuje na jeho převedení.

2.1 Původní informační systém SCCS

Superpočítač Anselm byl spuštěn v roce 2013. Po jeho spuštění vznikly požadavky na vedení a správu výpočetních zdrojů. Bylo potřeba vytvořit postupy, které by tyto požadavky podchytili a automatizovaly. Především z časových důvodů nebylo možné vyvíjet podpůrný software a často se využívaly existující nástroje, které nebyly k danému účelu určeny a přizpůsobeny. Postupem času již nebylo možné spravovat superpočítačové centrum předešlými postupy a vznikl IS SCCS zaměřující se na administraci projektů, uživatelů a přístupových práv.

Jelikož se často měnily a vznikaly nové požadavky na procesy IS spolu s požadavkem na jeho rychlou implementaci, byl za metodiku vývoje zvolen agilní FDD neboli vývoj řízený vlastnostmi. V první fázi vývoje byl vytvořen doménový model s vysokou úrovní abstrakce, popisující základ prvotních požadavků na informační systém. Díky doménovému modelu se dokážeme vyhnout problémům se spoluprací mezi různými prvky systému. Po vytvoření zmíněného modelu byl systém vyvíjen iterativně v přibližných 2 týdenních cyklech.

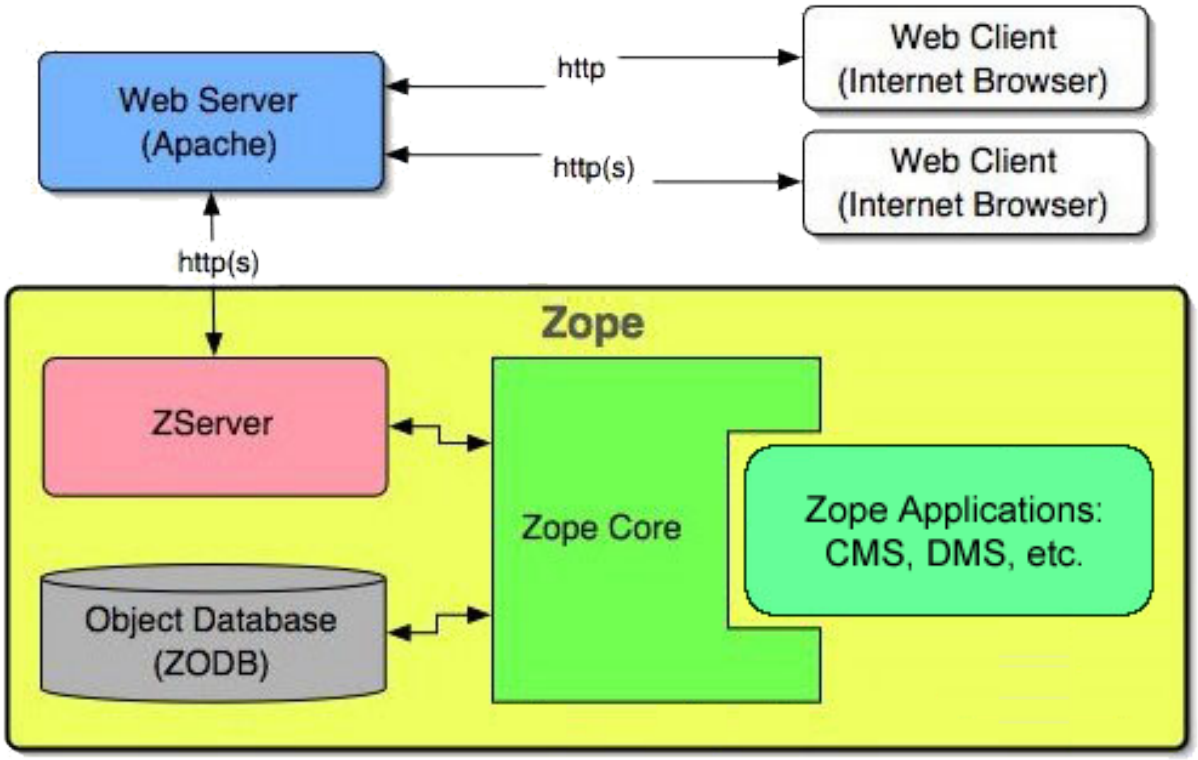
Pro vývoj aplikace systému SCCS byl zvolen software **Plone**, který je určen pro tvorbu CMS systémů. Běžně se využívá pro tvorbu intranetových webových portálů a byl vybrán především pro možnost využití velkého množství již existujících doplňků neboli tzv. *add-ons*, jako je například navázaní spojení s interní databází LDAP, správa webového obsahu nebo nastavení přístupových práv uživatelům.

Plone je postaven na aplikačním serveru Zope, který usnadňuje vývojáři práci s detaily vývoje aplikace jako je perzistence dat, jejich integrita a řízení přístupových práv. Ačkoliv se jedná o objektově orientovanou platformu, která poskytuje zaobalení základních operací pro práci s databází IS, nebyla tato vlastnost plně využita. Pro získávání a manipulaci dat pro jakékoliv procesy, metody, formuláře či přehledové stránky IS byly vytvořeny konkrétní SQL metody. Ty prováděly požadované úkony se záznamy nebo navracely získané kolekce dat zpět pohledům využívajícího šablonového jazyka TAL pro vykreslení dat do HTML stránek.

Vytváření a práce s objekty Zope se děje pomocí webového rozhraní ZMI. Každý objekt je uložen ve stromové hierarchii objektů a je možné jej skrze toto rozhraní konfigurovat. Například objekt reprezentující **databázové spojení** poskytuje pohledy, které umožňují upravit hodnoty konektorů spojení nebo nastavené spojení otestovat. Všechny objekty mají také pohled pro nastavení zabezpečení, které umožňuje spravovat jejich individuální nastavení přístupových práv.

Objekty Zope jsou uloženy v databázi transakčních objektů, známou jako ZODB. Jakýkoliv požadavek na web je považován za samostatnou transakci ze strany ZODB. Pokud dojde při běhu aplikace k chybě, budou všechny změny provedené během zpracování požadavku automaticky vráceny zpět. ZODB rovněž poskytuje vícestupňové vrácení změn zpět.

Strukturu a provázání základních komponent Zope, na kterém je postaven Plone, si můžeme spolu s jejich popisem ukázat na následujícím obrázku 1.



Obrázek 1: Základní komponenty Zope aplikačního serveru [1]

- **Web Server**
  Můžeme využít webových serverů jako Apache nebo Microsoft IIS, které zpracovávají požadavky před zasláním na webový server Zope. Webový server napomáhá zpracovávat příchozí požadavky, rozšiřovat jeho schopnosti (např. ukončit připojení HTTPS) a ukládat do paměti obsah poskytnutý softwarem Zope.

- **ZServer**
  Zope obsahuje integrovaný webový server, který poskytuje obsah aplikace.

- **Zope Core**
  Jedná se o jádro, které koordinuje činnost Zope a řídí rozhraní pro správu obsahu a jeho objektovou databázi.

- **Object Database**
  Databáze pro uložení Zope objektů ZODB.

  Všechny informace nemusí být ukládány v této databázi. Můžeme také spolupracovat s dalšími relačními databázemi, včetně Oracle, PostgreSQL nebo MySQL.

  Také můžeme pracovat s dokumenty a jinými soubory uloženými na souborovém systému serveru.

- **Zope Applications**
  Jedná se o předpřipravené doplňky pro Zope aplikace instalované do souborového systému serveru. Technicky jsou to běžné balíčky programovacího jazyka Python. Jedním z balíčků je také samotný Plone CMS systém.

V počátcích byl vývoj díky zvolené technologii a metodice velice rychlý a bylo možné podchytit nejdůležitější pracovní postupy IS. V průběhu času spolu s rostoucími požadavky na systém se stával vývoj obtížným, špatně řiditelným a složitým. Jednotlivá úskalí systému je popsána dále.

2.1.1 Použité technologie

- **Framework**
  Plone CMS
  Plone je redakční systém, který pomáhá zefektivnit správu a organizaci dokumentů. Plone se používá zejména pro tvorbu intranetových a veřejných webových portálů, umožňující běžným uživatelům vytvářet a udržovat informace na internetových stránkách nebo intranetu pouze za pomoci webového prohlížeče. Plone využívá služeb objektové infrastruktury Zope, která je stejně jako Plone napsána v jazyce Python [2].

- **Aplikační server**
  Zope Application Server
  Open-source aplikační server, který ve své transakční objektové databázi obsahuje mimo

vlastního webového obsahu skripty psané v jazyce Python, dynamické šablony pro generování HTML výstupu a další. Administrace aplikačního serveru probíhá skrze webové rozhraní, které umožňuje úpravy obsahu ve vestavěném textovém editoru. Aplikační server je možné rozšiřovat tzv. produkty, které základnímu aplikačnímu serveru zpřístupní různé funkcionality [3].

- **Databázový server**
  MySQL
  MySQL je volně dostupný systém pro správu relačních databází *RDBMS*, který používá jazyk strukturovaných dotazů *SQL* [4].

  ZODB
  Plone používá databázi objektů *ZODB* pro ukládání dat. Tato databáze může jednat nezávisle na procesu seskupeném přes síť nebo přes jiný databázový server. *ZODB* ukládá libovolný Python objekt s jakýmikoliv atributy. Není třeba vytvářet schéma databáze ani popisy tabulek, jako u systémů relačních databází [5].

  ZODB využívá jazyka *ZSQL*, jehož metody poskytují přístup k relační a externí databázi od společnosti *Zope*. Aplikace *Zope* jsou postaveny na objektově orientovaných databázích. Všechna data *Zope* jsou uložena v této databázi objektů. Databázové dotazy a příkazy mohou být použity k publikování relačních dat, shromažďování a ukládání dat v relačních databázích a vytváření složitých databázových aplikací [6].

- **Správa souborů**
  WebDAV
  Množina rozšíření HTTP protokolu, který poskytuje možnost správy souborů skrze tento protokol. *WebDAV* z webu vytváří zapisovatelné paměťové médium, ve kterém poskytuje prostředí pro vytváření, změnu a přesun souborů na server. Nad soubory v tomto paměťovém médiu je vytvořena vlastnost uzamykání souborů, která ovládá přístup k souborům a zabraňuje editaci jednoho souboru více vývojáři najednou [7].

  V prostředí Linux lze využít implementace *davfs2*, kdy můžeme k souborům uloženým v databázi aplikačního serveru Zope, kde je vše uloženo jako objekt, přistupovat z lokálního souborového systému.

- **Webový server**
  Apache HTTP Server
  Softwarový server, spuštěný v hardwarovém stroji připojeném do internetu zajišťující obsluhu prohlížečů jednotlivých návštěvníků. Mezi výhody Apache HTTP Serveru patří zejména dostupnost pro všechny hlavní platformy a také fakt, že je vyvíjen jako open-source [8].

  Webový server je oddělen od aplikačního serveru a tím zajišťuje případnou možnost škálování celého řešení napříč více webovými i aplikačními servery.

2.1.2 Výhody

Počáteční rychlý vývoj
Díky bohatým funkcím Plone byl počáteční vývoj na základě definovaného doménového modelu velmi rychlý. Prvotní verze systému splňovaly základní požadované funkcionality ulehčující práci administrátorům SCS týmu.

Přístupová práva
Zprovoznění správy přístupových práv je v Zope aplikacích možno řídit pomocí integrovaného frameworku **AccessControl**. Pomocí něj bylo definováno několik základních rolí, které jsou přiřazovány uživatelům. Oprávnění je automaticky kontrolováno pro každý pohled, metodu nebo skript, který je vyvolán příchozím HTTP požadavkem.

Integrace s uživateli LDAP databáze
Plone nabízí rozšiřující balíček pro integraci s protokolem LDAP, který plně podporuje všechny funkce a přístup tohoto protokolu k datům na adresářovém serveru. Konfigurace tohoto balíčku je poskytnuta pomocí uživatelského webového rozhraní ZMI. Uživatele a jejich skupiny uložené v adresářovém serveru je možné využít jako standardní uživatele Plone aplikace. Vytváření, mazání a úprava uživatelů nebo skupin je povolena v závislosti na konfiguraci spojení s LDAP. Díky tomuto rozšíření bylo v IS rovnou využito autentizace již existujících uživatelů superpočítačového centra.

Propojení různorodých databází
Propojení všech databází, které IS využívá, jako MySQL, ZODB a LDAP uživatele, přináší možnost využití ZSQL jazyka poskytující přístup k relačním a externím databázím. Tímto jazykem jsme schopni v portálu agregovat a zobrazovat informace napříč různými databázemi.

2.1.3 Nevýhody

IS založen na CMS
Ačkoliv prvotní verze systému vznikaly velice rychle, následný vývoj komplexnějších funkcí byl obtížný a často docházelo k úpravám zdrojového kódu samotného frameworku Plone CMS. To mělo za následky složitý vývoj, množinu vznikajících závislostí napříč moduly, nemožnost využití postupů oficiální dokumentace a také upravené nasazení aplikace či její aktualizace. Navíc CMS systémy jsou primárně určeny pro publikování článků, vytváření galerií a častou změnu obsahu. V tomto IS jsou jednou vytvořené agendy a jejich šablony opakovaně využívány pro zobrazení předaných dat z databáze.

Vývojové prostředí
Vytvoření vývojového prostředí pro otestování aktualizací aplikace, vývoj nových funkcí či rychlé

začlenění nového vývojáře do týmu nebylo nijak automatizováno. K dosažení vytvoření tohoto prostředí muselo být provedeno několik kroků.

- **Instalace portálu**
  Je potřeba vytvořit novou instalaci portálu v cílovém umístění pomocí Plone instalátoru. Před samotnou instalací portálu je potřeba zajistit instalaci všech potřebných knihoven.

- **Kopie složky zdrojových kódů**
  Z produkčního IS SCCS je potřeba zkopírovat složku *src*, která obsahuje všechny doplňky, které byly pro portál vytvořeny vývojáři.

- **Kopie objektové databáze ZODB**
  Dále je potřeba z produkční instance IS zkopírovat soubor *Data.fs*, který obsahuje ZODB databázi portálu. je potřeba podotknout, že heslo administrátora je uloženo v tomto souboru, tudíž heslo zadané během instalace vývojového prostředí již není platné. Dále musí být vytvořena kopie adresáře *var/blobstorage*, která obsahuje objekty typu BLOB databáze ZODB.

- **Kopie předpisu pro sestavení**
  Pro sestavení vývojové instance IS je potřeba vytvoření předpisu, který popisuje jeho jednotlivé kroky, využité doplňky, způsob jejich stažení či kontrolu jejich verzí. Předpis opět zkopírujeme z produkčního IS.

- **Migrace databázového schématu**
  V poslední řadě je potřeba provést export schématu relační databáze MySQL a aplikovat jej na naší nově instalované instanci.

Po této přípravě nastavení vývojového prostředí se spustí skript pro sestavení portálu **buildout** a pomocí webového rozhraní ZMI nastavíme potřebné konfigurační soubory.

Systém správy verzí

Jedno z úskalí IS SCCS bylo, že aplikace sice byla vedena verzovacím systémem Git, ale nebyl jím řízen vývoj. To je zapříčiněno především uložením objektů v databázi ZODB. Objekty vytvořené skrze ZMI jsou uloženy v souboru *Data.fs* jako datový typ BLOB. Nejsme tedy schopni řídit změny provedené tímto způsobem. Z toho důvodu je problém různé verze mezi vývojáři distribuovat a ujistit se, že se nacházejí ve stejném stavu.

Git repozitář podchycoval jen určitou skupinu změn, jako je např. složka *src* obsahující kódy vyvíjených doplňků. V Git repozitáři nebyly vedeny ani verze nástrojů samotného portálu. Tyto informace byly dostupné pouze na produkční verzi IS.

Vývoj v týmu

Změny, které vývojáři učinili ve svém vývojovém prostředí, museli často ručně vkládat do produkčního IS skrze webové rozhraní. Tím nebylo možné jednoduše vyřešit vznikající konflikty, pokud více vývojářů upravovalo stejný soubor.

Migrace databázového schématu

Dalším problémem, který v čase gradoval, bylo udržení verze relačního databázového schématu mezi vývojáři a jeho distribucí na produkční portál.

Záviselo pouze na dohodě, jak si SQL dotazy se změnami schématu budou vývojáři mezi sebou udržovat. Pro tento účel byl vytvořen adresář, kde byly jednotlivé migrační soubory uloženy. Historie aplikace změn nebyla v databázi udržována. Nebyli jsme schopni jednoduše zjistit, jaká změna byla naposled provedena.

Aktualizace IS

Pokud byla vyvinuta nová verze systému poskytující novou funkcionalitu nebo jakékoliv úpravy, které bylo potřeba uvést do provozu, muselo být provedeno několik kroků, které nebyly automatizovány.

- **Odstavení aplikace**
  IS bylo potřeba nastavit do stavu údržby. V tomto stavu nejsou služby ani rozhraní IS přístupné běžným uživatelům, ale pouze administrátorům.
- **Změny databázového schématu**
  Pokud nová verze IS vyžadovala aktualizaci relačního databázového schématu, bylo potřeba ručně aplikovat soubory obsahující SQL dotazy k tomu určené.
- **Zavedení změn**
  Některé změny jako vytvořený balíček bylo jednoduché naimportovat za pomocí Git repozitáře. Ovšem ostatní nastavení, objekty a také skripty uložené v objektové databázi ZODB musely být ručně vytvořeny skrze správu ZMI.
- **Sestavení a spuštění portálu**
  Pro portál se vyvolá akce *buildout*, která konfiguruje a instaluje rozšiřující balíčky a součásti IS. Portál je následně po sestavení spuštěn v běžném režimu.

Upravené chování původního frameworku

Nástavba Plone pro aplikační server Zope byla často přizpůsobována k vykonání práce, ke které není primárně určena. Tyto modifikace nebyly vývojáři dokumentovány, a proto některé prováděné změny způsobovaly problém i když se na první pohled zdálo, že nemohou nic ovlivnit.

Integrované API a RSS kanály

Původní IS poskytoval XML-RPC API a také RSS kanály pro poskytování informací jak uživatelům, tak i různým extérním aplikacím.

Plone poskytuje užitečnou funkci nazvanou **syndikace** umožňující publikování obsahu složek Plone CMS do RSS kanálu. Tyto kanály poskytovaly MOTD neboli zprávy dne a data pro grafy jako vytížení superpočítače, využití výpočetního času organizacemi apod.

API XML-RPC bylo využito hlavně pro poskytnutí služby ke kontrole využitých zdrojů projektů konkrétního uživatele. Dále bylo díky API možno získat informace o uživatelích uložených v databázi LDAP a licence poskytované uživatelům superpočítačového centra. Vytvoření a nasazení tohoto API bylo velice jednoduché a rychlé. Zope poskytuje integrovaný XML-RPC server kde jakýkoliv objekt může odpovídat na požadavky protokolu HTTP. Stačilo pouze vytvořit potřebné metody a o komunikaci a běh serveru se již stará samotný Zope. Velkou výhodou je možnost využití již existujícího autentizačního modulu.

Největší omezení původního API a RSS kanálu bylo, že služby nebyly poskytovány, pokud aplikace byla vypnutá, procházela údržbou, byla nasazována nová verze systému nebo se nedokázala zotavit z nějaké nestandardní chyby. Také zde nebyla možnost jednoduchého nastavení dalšího klienta, na kterého by byly dotazy přesměrovány v případě pádu aplikace.

Testování

Ačkoliv Plone poskytuje různé typy automatizovaných testů, tak nebyly nikdy využity a implementovány.

2.2 Nový informační systém SCS

Nemožnost dalšího rozvoje původního systému SCCS vedla k rozhodnutí vytvoření nové generace informačního systému. Toto rozhodnutí bylo zapříčiněno mnoha faktory, jako odchod vývojářů původního IS, nesdílené know-how, nezadokumentované postupy, nestandardní úpravy frameworku a ostatními nevýhodami, které byly popsány v předchozí kapitole. Vývoj nového systému tak byl zahájen na počátku roku 2017.

Největším omezením při vývoji nového řešení IS tkvělo v tom, že nebyla možnost zcela nového návrhu celého systému. Muselo se navázat na již zavedené postupy, databázové schéma, na kterém je navázáno několik externích systémů a poskytované API spolu s RSS kanály původního IS využívaného externími službami.

Z důvodů požadavků na co nejrychlejší možné uvedení nového systému do provozu byla zvolena agilní metodika vývoje zvaná **extrémní programování**. Tato metodika vývoje si zakládá na časté komunikaci mezi programátory a zákazníkem, jednoduchosti, kdy programujeme pouze to, co je potřeba ke splnění aktuálních požadavků, častém dodávání produktu v krátkých cyklech a neustálou revizí kódu.

Nový IS je tvořen open-source frameworkem Ruby On Rails, určeného pro vývoj webových aplikací v jazyce Ruby. Framework vznikl v prosinci roku 2005 programátorem Davidem

Hanssonem. Je navržen tak, aby usnadnil programování webových aplikací na základě obecných předpokladů vývoje pro web.

Rails si zakládá zejména na dvou principech a to *konvence před konfigurací* a *DRY*.

- **Konvence před konfigurací**
  Vývoj pomocí Rails se nás snaží vést k jakémusi „*nejlepšímu přístupu*“. Pokud tedy dodržujeme definované konvence jako názvy tabulek, relace tříd apod. Rails předpokládají co a jak chceme udělat, místo aby jsme specifikovaly každou maličkost v konfiguračních souborech.

- **DRY**
  Jedná se o způsob rozvoje softwaru, který snižuje opakování informací nebo kódu. Vede nás tedy ke psaní znovupoužitelného kódu. Pro dosáhnutí tohoto principu Rails využívá částečných pohledů k odstranění duplicit zobrazení, nebo pomocných filtrů v třídách řídící komunikaci uživatele s aplikací, kdy můžeme definovat společné akce, které se mají vykonat před zahájením metod této třídy.

Pro dosažení zmíněných dvou principů je framework Rails postaven na architektuře MVC. Adresářová struktura aplikace odpovídá této architektuře, kde nalezneme podsložky *models*, *views* a *controllers*. Mezi výhody této architektury patří oddělení aplikační logiky od uživatelského rozhraní, znovupoužitelnost kódu (princip DRY) a přehledná struktura kódu aplikace usnadňující její údržbu.

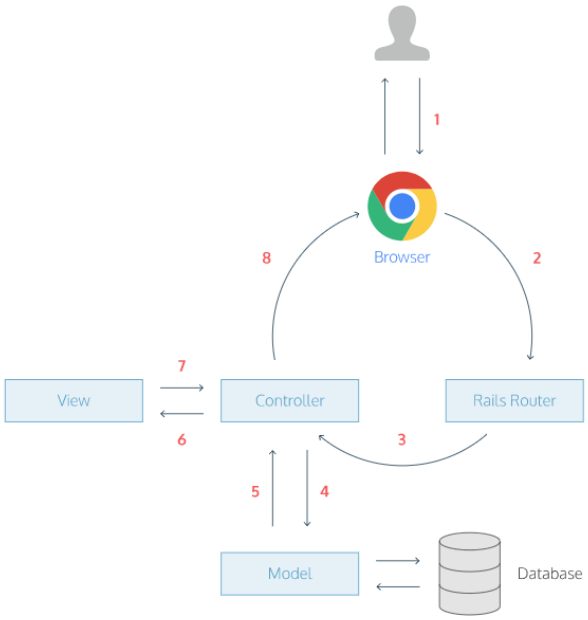
- **Model**
  Model reprezentuje data aplikace a pravidla pro práci s nimi. Jakýkoliv příkaz uživatele prochází touto vrstvou. Modely jsou primárně využívány pro interakci s příslušnou databázovou tabulkou. ve většině případů odpovídá jedna tabulka v databázi jednomu modelu aplikace. Modely obsahují většinu aplikační logiky.

  Práce s modely v Rails nás odstiňuje od nutnosti psaní *SQL* dotazů a prací s databázovou vrstvou. Také jsme díky modelům nezávislí na databázové vrstvě. Toto chování je dosaženo využitím programovací techniky ORM, která se stará o konverzi dat mezi relační databází a objekty, se kterými pracujeme v objektově orientovaném jazyce. Dostáváme tak unifikovaný přístup k libovolnému objektu aplikace. Pro dosažení práce s ORM je implementován návrhový vzor *Active Record*, který nám všechny tyto aspekty zajišťuje. V AR je nejvíce využit princip *konvence před konfigurací*, kdy automaticky rozpozná konfiguraci, pokud je správně dodržená konvence Rails.

- **View**
  Pohledy reprezentují uživatelské rozhraní aplikace. Určují, jak uživatel interaguje s aplikací a jak jsou mu prezentována data. Pohledy jsou obvykle HTML soubory s vloženými značkami Ruby kódu, které provádí úkony pro vykreslení HTML stránky spolu s požadovanými daty.

- **Controller**
  Řídící jednotky slouží ke zpracování příchozích požadavků z webového prohlížeče, získání dat z modelů a odesílání těchto dat pohledům, které budou uživateli zobrazeny.

Znázornění cyklu zpracování požadavku uživatele v IS, vyvinutém pomocí Rails, můžeme vidět na obrázku 2.



Obrázek 2: Zpracování příchozího požadavku frameworkem Rails [9]

1. Uživatel otevře prohlížeč, zadá URL adresu a potvrdí akci, čímž vytvoří požadavek.
2. Požadavek prochází směrovací tabulkou.
3. Směrovač namapuje URL správné řídící jednotce a její příslušné akci ke zpracování požadavku.
4. Akce řídící jednotky obdrží požadavek a dotáže se odpovídajícího modelu na načtení dat z databáze.
5. Model navrátí potřebný seznam dat řídící jednotce.
6. Akce řídící jednotky přenese data relevantnímu pohledu.
7. Pohled vykreslí HTML stránku s předanými daty.
8. Řídící jednotka zašle zpět vykreslenou HTML stránku prohlížeči, kde se načte a zobrazí uživateli.

2.2.1 Použité technologie

- **Framework** - Ruby On rails
  Ruby On Rails, zkráceně Rails, je multiplatformní open-source framework pro pohodlné a rychlé vytváření moderních webových aplikací. Dnes jej používají společnosti jako je Apple, Oakley, The New York Times, Twitter, Electronic Arts, Github nebo Yellow Pages [10].

- **Aplikační server** - Puma
  Puma je aplikační server, který umožňuje aplikaci Rails zpracovávat žádosti souběžně. Vzhledem k tomu, že Puma není navržena tak, aby byla přístupná přímo uživatelům, využijeme webového serveru Nginx jako reverzní proxy, který bude zpracovávat požadavky mezi uživateli a Rails aplikací [11].

- **Databázový server** - MariaDB
  Databázový server MariaDB je jedním z nejoblíbenějších databázových serverů na světě. je vytvořen původními vývojáři MySQL, kteří zaručují, že databáze zůstane open-source. Mezi významné uživatele patří Wikipedia, WordPress.com a Google. MariaDB přeměňuje data na strukturované informace v široké škále aplikací, od bankovnictví po internetové stránky. Jedná se o zdokonalenou náhradu za databázový server MySQL. MariaDB se používá, protože je rychlá, škálovatelná a robustní, s bohatým výběrem paměťových úložišť, pluginů a mnoha dalších nástrojů, díky nimž je univerzální pro celou řadu případů užití [12].

- **Webový server** - Nginx
  Nginx je bezplatný, open-source, výkonný HTTP server a reverzní proxy, stejně jako protokol proxy IMAP/POP3. Je znám svou vysokou výkonností, stabilitou, bohatou sadou funkcí, jednoduchou konfigurací a nízkou spotřebou zdrojů.

  Na rozdíl od tradičních serverů se NGINX nespoléhá na podprocesy, které by mohly zpracovávat požadavky. Místo toho používá mnohem škálovatelnější událostně řízenou (asynchronní) architekturu. Tato architektura používá malé, ale hlavně předvídatelné množství paměti pod zatížením. Dokonce i když neočekáváte, že budete zpracovávat tisíce současných požadavků, můžete stále těžit z vysokého výkonu a malé paměťové náročnosti [13]. Nginx oproti webovému serveru Apache spotřebuje na jedno spojení méně paměti, takže dokáže obsloužit více uživatelů za využití stejných prostředků.

2.2.2 Systém správy verzí

Vývoj zdrojového kódu SCS IS je řízen systémem správy verzí Git, který ukládá jednotlivé změny kódu a je možné se k těmto změnám kdykoliv vrátit. Návrat k předchozí verzi kódu je během vývoje nezbytností.

Git je v současnosti nejrozšířenější systém správy verzí díky své rychlosti, škálovatelnosti a širokým možnostem využití. Jako centrální server pro repozitáře je využit open-source software

GitLab, který podporuje vývoj neveřejných projektů. Umožňuje také správu řízení přístupu čímž udržuje náš kód v bezpečí. Webové rozhraní poskytuje kontrolu a možnosti sloučení kódu. Každému projektu můžeme také vytvořit dokumentaci, nastavit kontinuální integrace apod.

Vlastnosti SCS IS jsou rozvíjeny pomocí tzv. *issue*, kterými jsou popsány nové funkcionality, vylepšení a také vzniklé chyby v systému. Z těchto požadavků jsou dále vytvořeny vývojové větve, které jsou po dokončení vývojářem navrhnuty na sloučení do hlavní vývojové větve *master*, kterou je možné aplikovat na produkční IS. Návrhy na sloučení vývojových větví s hlavní větví *master* jsou vždy přiřazovány jinému vývojáři, který nad změnami provádí kontrolu. Pokud je vše v pořádku, změny jsou akceptovány. Vývoj tedy podléhá tzv. kontrole dvojích očí.

2.2.3 Vývojové prostředí

Pro vytvoření a nastavení vývojového prostředí je využito nástroje **Vagrant**. Tento nástroj je určen pro budování kompletního vývojového prostředí. Jsme schopni vytvářet snadno použitelný pracovní postup se zaměřením na automatizaci, kdy Vagrant snižuje dobu nastavení vývojového prostředí za pomocí virtualizace.

Díky doplňku označovaného jako *provider plugin* můžeme vytvářet virtuální vývojové prostředí jednoho nebo více počítačů na hostitelské virtuální infrastruktuře **vSphere**. Tímto provázáním jsme schopni ze svého lokálního stroje, pomocí klonovacího předpisu Vagrant, toto virtuální vývojové prostřední do infrastruktury vytvořit a poté se k němu připojit.

Pro vytvoření vývojového prostředí byl vytvořen předpis, který provádí následující kroky:

1. **Instalace potřebného softwaru**
   - Git
   - Potřebné knihovny
   - Redis
     Open-source paměťová struktura dat v paměti, používaná jako databáze, mezipaměť a zprostředkovatele zpráv. Jedná se o *key-value* databázi což umožňuje svázat klíč s hodnotou.
   - RVM
     Nástroj, který umožňuje snadno instalovat, spravovat a pracovat s více prostředími programovacího jazyka Ruby.
   - Databázový server MariaDB
2. **Nastavení Databáze**
   Po instalaci databázového serveru MariaDB proběhne část předpisu, který databázový server nastaví.
3. **Instalace Ruby On Rails**
   Instalace Ruby On Rails je řízena nástrojem *RVM*. Díky tomuto nástroji jsme schopni

testovat Rails aplikaci oproti novým verzím. Testování je velmi snadné, jelikož můžeme rychle přepínat mezi více verzemi jazyka Ruby i frameworku Rails.

4. **Klon Git repozitáře**
   V téhle fázi je ze serveru GitLab naklonován repozitář *scs-ror* obsahující IS SCS. Po naklonování se nastaví proměnné prostředí pro SCS IS v souboru *.env*. Dále se vytvoří databáze pro prostředí vývoje a testování.

   V posledním kroku je spuštěn příkaz *bundle install*, který nainstaluje všechny *gems*, což jsou knihovny, doplňky či pluginy, uvedené v souboru *Gemfile* a k nim také nainstaluje všechny potřebné závislosti. **Bundler** nám zajišťuje nastavení stejného prostředí a balíčků napříč všemi prostředími využívajícího repozitáře *scs-ror*.

5. **Vložení vývojových dat**
   Fáze vytvoření dat pro vývoj prozatím není automatizována. Aby mohl vývojář data do aplikace vložit je potřeba, aby se na vytvořený virtuální stroj přihlásil, stáhl si Git repozitář *scs-test-data*, který byl pro tento účel vytvořen a data si podle postupu v *README* tohoto repozitáře vložil do databáze. *README* popisuje import *MysqlDump* souborů, což jsou logické zálohy obsahující sadu *SQL* příkazů, které lze spustit pro reprodukci původních definic databázových objektů a dat, do již vytvořeného databázového serveru.

Po provedení těchto kroků už stačí na vývojové instanci spustit příkaz *rake db:migrate*, který provede zatím neaplikované migrační skripty nad databází. Kolik migrací se provede závisí na tom, jak je aktuální repozitář *scs-test-data* zmíněný v posledním bodě předpisu, viz výše.

2.2.4 Testovací instance IS

Nesmírnou výhodou oproti vývoji původního SCCS IS je existence testovací instance. Veškeré provedené změny, které mají být aplikovány na produkční instanci IS jsou nejprve aplikovány na tuto testovací instanci. Aplikované změny jsou předvedeny a předány členům administrátorského týmu k otestování. Pokud nejsou změny vyhovující, jednoduše se dají systémem správy verzí Git navrátit zpět do původního stavu.

2.2.5 Kontinuální integrace

Pro kontinuální integraci je využit nástroj, který nabízí GitLab server a to **GitLab Continuous Integration & Deployment**. Nastavení CI je velice jednoduché, stačí v kořenovém adresáři projektu vytvořit soubor *.gitlab-ci.yml* pro definování úkonů CI a nakonfigurovat projekt pro použití služby **Runner**. Tato služba je použita pro spuštění úloh a navrácení jejich výsledku zpět na server Gitlab. Po nastavení předpisu pro CI a služby Runner každý *commit* nebo *push* spustí *pipeline* kontinuální integrace.

Runner je nastaven tak, aby využíval enginu **Docker**, který umožňuje používat předdefinované obrazy disků pro spouštění aplikací. ve spolupráci s CI, provádí každou úlohu v předdefinovaném

obrazu disku, který je při každém spuštění reinicializován. Je takto zajištěno reprodukovatelné a stabilní prostředí.

Kontinuální integrace pro SCS IS je rozdělena do třech fází a to *Setup*, *Test* a *Deploy*. Každá fáze kontinuální integrace je prováděna samostatně v definovaném pořadí a obsahuje své úlohy, které musí vykonat.

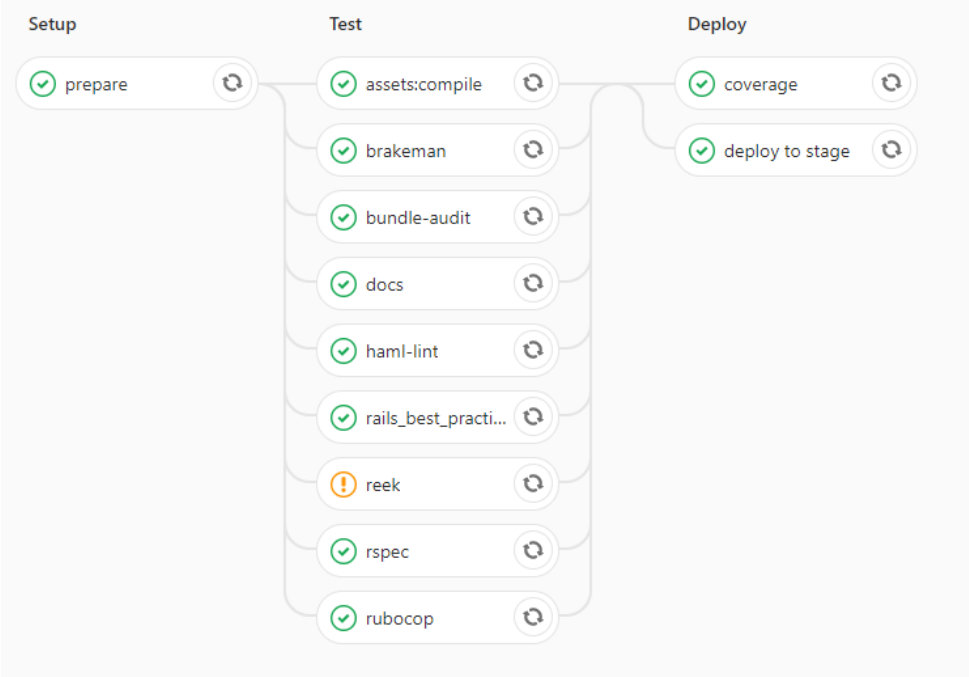
- **Setup** fáze slouží pro nastavení prostředí CI projektu scs-ror, nad kterým budou spouštěny testy. Tato fáze vykonává pouze jediný úkol s názvem *prepare*.
  - **Prepare**
    Úloha vykonává instalaci frameworku Rails spolu s knihovnami definovanými v souboru *Gemfile*. Instalace probíhá v módu *deployment*, která se využívá buď pro nasazení aplikace na produkční server nebo právě pro instalaci aplikace v CI.
- **Test** fáze slouží pro spuštění různých kontrol kódu a testů. je zde vykonáváno nejvíce úloh. Pro tyto úlohy již máme nakonfigurováno prostředí díky předešlé fázi *setup*.
  - **Sestavení aktiv - assets:compile**
    Provádí test sestavení manifestu aktiv tzv. *asset*. Sestavení aktiv provádí spojování, zmenšování nebo kompresi prostředků JavaScript a CSS. je zde schopnost zapisovat aktiva do jiných jazyků a pre-procesorů, jako jsou CoffeeScript, Sass a ERB. Umožňuje kombinaci aktiv aplikace s aktivy jiných balíčků. Sestavené aktiva jsou uložena v souborovém systému a webovým server je může přímo zobrazovat.
  - **Statická kontrola chyb zabezpečení - brakeman**
    Jedná se o nástroj pro detekci zranitelnosti, který je speciálně navržen pro aplikace Rails. Staticky analyzuje kód aplikace a hledá známé bezpečnostní problémy v jakékoli fázi vývoje.
  - **Kontrola balíčků - bundle-audit**
    Kontroluje zranitelné verze nainstalovaných gem balíčků a jejich zdroje. Upozorňuje také na jejich nové verze.
  - **Kontrola Markdown souborů - docs**
    Tato úloha provádí kontrolu Markdown souborů, jestli jsou vytvořeny podle správné konvence a stylu kódu.
  - **Kontrola Haml souborů - haml-lint**
    Napomáhá udržet HAML soubory čisté a čitelné. je integrován s nástrojem **RuboCop**, což přináší výkonný nástroj pro statickou analýzu Haml dokumentů.
  - **Rails nejlepší postupy - rails_best_practices**
    Je kódový nástroj pro řízení kvality kódu podle definovaných nejlepších postupů komunity Rails.

  - **Kontrola problémů zdrojového kódu - reek**
    Nástroj zkoumající třídy, moduly a metody a hlásí jakékoliv podezření, které najde. Podezření indikuje místa, kde může být kód špatně čitelný, udržovaný nebo vyvíjený. Hlásí také porušení principu DRY.
  - **Testy aplikace - rspec**
    RSpec je framework určený pro testování aplikací Rails, který je postaven na filozofii **BDD**, což je přístup k vývoji softwaru, který kombinuje vývoj řízený testy, doménově řízený design a plánování akceptačních testů.
  - **Statický kódový analyzátor - Rubocop**
    Ruby analyzátor statického kódu Rubocop prosazuje mnoho pokynů definovaných komunitou *Ruby Style Guide*. Prosazuje osvědčené postupy vývoje kódu tak, aby programátoři mohli psát kód, který mohou udržovat i jiní programátoři.

- Hlavním úkolem fáze **deploy** je nasazení změn na testovací instanci IS. Nasazení je závislé na předešlé fázi *test*. Pokud úlohy *test* fáze neproběhnou podle očekávání, není nasazení změn na testovací instanci provedeno.

  - **Pokrytí kódu - coverage**
    SimpleCov je nástroj analýzy pokrytí Ruby kódu rspec testy. Používá zabudovanou knihovnu *coverage* pro shromažďování dat o pokrytí kódu. Poskytuje knihovnu API pro filtrování, seskupování, sloučení, formátování a zobrazení těchto výsledků. Pokrytí kódu se v informačním systému nyní pohybujeme okolo 20%.
  - **Nasazení - deploy to stage**
    Tato úloha má na starost nasazení změn na testovací instanci. Děje se tak pouze, pokud jsou všechny závislosti splněny a jednotlivé předešlé úlohy byly ukončeny podle očekávání. Na testovací instanci se aplikují změny, které jsou provedeny v hlavní vývojové větvi *master*. Před nasazením na produkční instanci informačního systému probíhá kontrola testovacího webu vývojářem a pokud se změny zdají být v pořádku, mohou se aplikovat také na produkční instanci. Nasazení změn na produkční instanci provádí vývojář manuálně pomocí spuštění skriptu.

Souhrn fází a jejich úloh můžeme vidět na obrázku 3.



Obrázek 3: GitLab kontinuální integrace

2.2.6 Extérní API informačního systému

V původní verzi informačního systému SCCS bylo vytvořeno integrované XML-RPC API spolu s RSS kanály, které poskytovaly informace jak pro uživatele, tak i externí systémy. Oba moduly závislé na původním IS byly odděleny do samostatného REST API modulu. Díky oddělení je nové REST API nezávislé a schopné dodávat výsledky i během výpadku či údržby IS. Jediná závislost pro běh tohoto nástroje je spuštěný databázový server IS s aktivním připojením.

Projekt nese název SCS-API a jedná se o jednoduchý API server vytvořený pomocí frameworku **Flask**, který je označován za mikro framework, protože nevyžaduje žádné další nástroje nebo knihovny pro svůj běh. Neobsahuje žádnou databázovou abstrakční vrstvu, validaci formulářů ani žádné jiné součásti poskytující funkcionality třetích stran. na druhou stranu podporuje rozšíření pro přidávání funkcí, jako kdyby byly implementovány v samotném balíčku. K dispozici je např. ORM, nahrávání souborů či autentizační moduly.

Toto API plní několik úloh:

- **check-access**
  Adresa: POST /api/v1/check-access
  Služba, která kontroluje, zda má účet a související projekt přístup k výpočetním uzlům superpočítačového centra. Detail této metody nalezneme v kapitole 3.6 popisující zadání úlohy do *HPC* systému.

- **dedicated-time**
  Adresa: GET /api/v1/dedicated-time/(cluster_type)

Navrací seznam časů určených pro údržbu *HPC* systému. Během údržby není možné využívat výpočetní uzly superpočítačového centra.

- **it4ifree**
  Adresa: POST /api/v1/it4ifree/(login)
  Služba ke kontrole využitých zdrojů projektů, na kterých se účet uživatele účastní. Výstup poskytuje data rozdělená do dvou částí:
  - **me** zobrazuje dostupné zdroje projektu spolu s využitými jádrohodinami uživatele, přiřazeného k tomuto projektu.
  - **me_as_pi** poskytuje údaje z projektů, kde účet uživatele vystupuje v roli hlavního řešitele PI. K projektům jsou zobrazeni jeho uživatelé a jejich využité zdroje.

- **motd**
  Adresa: GET /api/v1/motd/(category)
  Zobrazuje zprávy dne.

- **ping**
  Adresa: GET /api/v1/ping
  Služba testující spojení aplikace a nástroje SCS-API. V případě aktivního spojení navrací zprávu *pong*.

- **version**
  Adresa: GET /api/v1/version
  Navrací základní informace ohledně nástroje *SCS-API*.

Všechny úlohy poskytují návratové hodnoty ve formátu JSON.
Informace pro většinu metod jsou získávány z relační databáze informačního systému SCS. Navázaní spojení s databázovým serverem a vykonávání dotazů nad ním je spravováno pomocí balíčku nástrojů **SqlAlchemy**, který poskytuje plnou sadu známých vzorů, pro perzistentní vrstvu, navržených pro efektivní a výkonný přístup k databázi, přizpůsobené skriptovacímu jazyku Python.

2.2.7 Testování

Testovaní informačního systému je řízeno nástrojem **RSpec**, využívající vývojový model označovaný jako BDD neboli vývoj řízený požadavky na chování pro jazyk Ruby. Obecně komunita Rails vede vývojáře k psaní aplikace stylem TDD tedy vývoje řízeného testy. Nejprve píšeme testy aplikace. Pokud je spustíme, tak testy musí selhat. Dále je psán dostatečný kód, který po opětovném spuštění testů projde. Výsledný kód pak prochází refaktorizací.

Nástroj *RSpec* se skládá z několika knihoven [14]:

- **rspec-core**
  Poskytuje strukturu pro psaní spustitelných příkladů, příkazovou řádku s mnoha funkcemi, flexibilní a přizpůsobitelné výstupy.

- **rspec-expectations**
  Poskytuje čitelné API rozhraní, které umožňuje vyjádřit očekávané výsledky objektu v testovacím případu.

- **rspec-mocks**
  Napomáhá řídit kontext v testovacích případech tím, že umožňuje nastavit očekávané návratové hodnoty, falešné implementace metod a také očekávané zprávy přijímané objektem.

- **rspec-rails**
  Podporuje použití knihovny RSpec pro testování Rails aplikací namísto vestavěného testovacího frameworku *minitest* integrovaného ve frameworku Rails. Rozšiřuje testovací systém pro podporu testování požadavků, řídících jednotek, modelů, pohledů, pomocníků, odesílatelů e-mailů a směrovačů.

Ačkoliv RSpec poskytuje širokou škálu testovacích případů, jsou v současnosti pokryty především přístupová práva uživatelů k jednotlivým sekcím IS, jednotlivé modely a jejich validace. Testy těchto sekcí jsou pokryty za pomocí testů řídících jednotek aplikace.

Celkové pokrytí kódu testovacími případy se nyní pohybuje okolo 20%.

Z důvodů rapidního vývoje a faktu, že do systému nyní přistupuje pouze administrátorský tým superpočítačového centra, byla tvorba testovacích případů odložena do pozdější iterace implementace IS.

2.2.8 Přístupová práva

Oproti původnímu IS využívajícího integrovanou správu přístupových práv frameworku **Zope** spolu s návazností na protokol *LDAP*, bylo využito extérního doplňku pro autentizaci uživatelů.

Jedná se o knihovnu **Devise**, která je postaven na modulárním konceptu, kdy můžeme využít pouze potřebných vlastností pro konkrétní účely [15]:

- **Databázová ověřitelnost - Database Authenticatable**
  Šifruje a ukládá heslo do databáze pro ověření uživatele během přihlášení. Ověřování lze provést prostřednictvím POST požadavku nebo také pomocí *HTTP Basic Authentication*.

- **Omniauthable**
  Přidává podporu pro **OmniAuth**, což je knihovna, která standardizuje ověřování uživatelů více poskytovateli pro webové aplikace. Každý vývojář může vytvořit strategie pro OmniAuth, které mohou autentizovat uživatele prostřednictvím rozdílných systémů.

V současnosti probíhá ověřování uživatelů vůči internímu serveru Gitlab *code.it4i.cz*, který je přístupný všem uživatelům, kteří jsou vedeni v interní databázi protokolu *LDAP* superpočítačového centra.

- **Potvrzovatelnost - Confirmable**
  Odesílá e-maily s pokyny k potvrzení a ověřuje, zda je účet již potvrzen během přihlašování.

- **Obnovitelnost - Recoverable**
  Obnovuje uživatelské heslo a zasílá instrukce potřebné k obnovení.

- **Registrovatelnost - Registerable**
  Zpracovává registraci uživatelů prostřednictvím registračního procesu a umožňuje uživatelům úpravu či smazání účtu.

- **Zapamatovatelnost - Rememberable**
  Spravuje generování a mazání autentizačního tokenu pro zapamatování přihlášení uživatele skrze uložený cookie soubor.

- **Sledovatelnost - Trackable**
  Sleduje čas, IP adresu a počet přihlášení uživatele.

- **Vypršitelnost - Timeoutable**
  Ukončí relaci spojení, která již není aktivní.

- **Ověřitelnost - Validatable**
  Poskytuje ověření e-mailu a hesla. Můžete definovat také vlastní ověření.

- **Uzamykatelnost - Lockable**
  Uzamkne účet po definovaném počtu neúspěšných pokusů o přihlášení. Může odemknout účet e-mailem nebo po uplynutí určitého času.

K uživatelům, kteří jsou ověřováni pomocí modulu Devise, byly nadále vytvořeny tři základní role pro omezení přístupu k jednotlivým agendám a to:

- **Admin**
  Uživatelé s touto rolí jsou oprávněni v systému provádět veškeré dostupné akce, jako je úprava, mazání a vytváření nových záznamů. Tato role slouží pro administrátorský tým superpočítačového centra.

- **Watcher**
  Tato role slouží pouze pro zobrazování a exportování informací v systému. Uživatele s touto přiřazenou rolí nemohou provádět žádné úpravy nad záznamy. Tuto roli mohou přiřazovat uživatelům pouze administrátoři.

- **Owner**
  Tato role je automaticky přiřazována uživatelům, kteří se do informačního systému prvně přihlásí skrze ověřovací server Gitlab. Oprávnění této role prozatím nejsou dokončena. Uživatelé s touto rolí nemohou zobrazovat ani upravovat jakékoliv agendy IS. V budoucnu by měla tato role sloužit pro uživatelé HPC systému pro zobrazování informací o jejich úlohách, projektech a provádění žádostí např. jako je žádost o přidání fronty k projektu.

Je plánováno přístupová práva propojit s uživateli, kteří jsou vedeni v interní databázi LDAP. Akce jako vložení, úprava a uzamykání účtů uživatelů nad touto databází mají být řízeny pomocí vyvíjeného IS pro usnadnění práce administrátorského týmu.

2.2.9 Cron úlohy

Pro nastavení **cron** úloh, což je softwarový démon spouštějící příkazy či procesy v definovaném čase, byl do IS zaveden balíček **Sidekiq**, který nám umožňuje tyto úlohy vytvářet. Sidekiq poskytuje jednoduché, efektivní a asynchronní zpracování dlouhodobých či hromadných úloh na pozadí pro jazyk Ruby. Využívá vlákna, které zpracovávají úlohy současně ve stejném procesu.

Vytvořené cron úlohy můžeme rozdělit do několika kategorií a to **kontroly**, **e-maily** a **procedury** systému.

- **Kontrola**
  V této kategorii se spouští systémové kontroly, které při zjištění chyby notifikují o nápravu buď autora úlohy nebo definované uživatele. Kontroluje se např. jestli existuje více uživatelských účtů na jedno uživatelské jméno, skupiny VPN, které nemají žádné aktivní uživatele, uživatelé bez aktivních projektů, jejichž účty by měly být uzavřeny apod. Jedná se tedy především o kontroly anomálií dat vyskytujících se v aplikaci.

- **E-maily**
  Kategorie má na starost kontroly stavů, zdali má dojít k odeslání e-mailů uživatelům. E-maily se odesílají v případech jako oznámení o brzké expiraci projektu, notifikace o běžícím projektu nevyužívající výpočetní čas nebo expiraci uživatelského účtu. V úlohách je definována kontrola, zda byl e-mail odeslán, aby nedošlo k vícenásobnému odeslání téhož e-mailu.

- **Procedury**
  Úlohy této kategorie slouží pro nastavování hodnot, údržbě systému atd. Informace o provedených akcích nejsou nikomu odesílány. Vykonává úlohy jako generování tokenu novým uživatelům, deaktivace projektů, přepočítává měsíční souhrny využité alokace projekty a podobné servisní úlohy.

2.3 Nasazení SCS IS do běžného provozu

Nasazení nového IS SCS, vytvořeného ve frameworku Rails, který nahrazuje původní verzi IS postaveného na frameworku Plone, bylo původně navrženo procesem nazývaným tzv. *drop-in* nahrazením. Tento proces má za úkol nahradit jeden softwarový komponent jiným. Musely být převedeny všechny funkcionality, vlastnosti, data a návaznosti extérních softwarů do nového IS tak, aby nedošlo k narušení žádných poskytovaných služeb a uživatelé využívající tento systém s ním mohli pracovat očekávaným způsobem.

Pro zachování všech zmíněných vlastností bylo potřeba provést následující:

- **Vytvoření prostředí**
  Pro IS bylo vytvořeno a nakonfigurováno prostředí umožňující jeho provoz ve VMware infrastruktuře, která umožňuje spravovat, optimalizovat a transformovat systémy pomocí virtualizace.

- **Migrace dat**
  Byly vytvořeny migrační skripty pro převedení dat z původního informačního systému do lehce upraveného schématu relační databáze nového IS.

- **Nastavení portálu Extranet**
  Webovému portálu Extranet bylo nastaveno databázové spojení na novou databázi IS.

  V tomto kroku vznikl problém, kdy portál Extranet nebyl schopen komunikovat s upraveným databázových schématem nového informačního systému, a to z důvodů změněných názvů tabulek, atributů apod.

  Portál Extranet se nachází také ve stavu, kdy jej nejsme schopni dále rozvíjet nebo upravovat. Přepsání všech skriptů portálu pro využívání nového schématu a následné otestování by bylo velmi složité a časově náročné. Proto jsou pro portál Extranet v novém databázovém systému vytvořeny pohledy simulující původní schéma databáze, kterou Extranet využívá. Z pohledu Extranet portálu se tedy zdá, že nic nebylo změněno a může fungovat stále stejným způsobem bez jakéhokoliv zásahu do kódu.

- **Nastavení extérních služeb**
  Služby využívající původního API musely být přeprogramovány pro využívání nového SCS-API serveru. Toto API poskytuje data ve formátu JSON. Služby očekávající návratové hodnoty API ve formátu XML musely změnit způsob zpracování návratových hodnot.

  Komunikaci s SCS-API využívají:

  – *Přihlašovací uzly HPC systému*
    Tyto uzly po přihlášení uživatele pomocí terminálu zobrazují díky API informace o zprávách dne nebo plánovaných výpadcích. Dále je na těchto uzlech definovaná metoda *it4ifree* využívající API pro zobrazení spotřeby výpočetního času přihlášenému uživateli.

– *Plánovač PBS*
Plánovač PBS využívá API pro získání informace, zda uživatel žádající o zařazení úlohy do systému HPC má na tuto akci oprávnění.

- **Zavedení LDAP uživatelů**
Data o uživatelích, kteří jsou vedeni v databázi LDAP, jsou vkládány do relační databáze nového IS z důvodů možné selekce těchto uživatelů, přiřazování k projektům a kontrole jejich úloh. Tyto data jsou aktualizovány na základě změny v databázi LDAP, která vyvolá příkazy pro změnu dat v databázi IS.

  V budoucnu by měl nový IS převzít roli LDAP správy uživatelů, kdy změny na straně databáze IS budou vyvolávat změny na LDAP databázi. Tím by administrátoři mohli spravovat uživatele superpočítačového centra pomocí webového rozhraní IS, namísto nynějších skriptů pro přidávání uživatelů, vytváření skupin nebo nastavování diskových kvót uživatele pro HPC systémy.

- **Evidence účtovacího času**
Pro uzly superpočítačového centra, které zpracovávají účtování výpočetních úloh uživatelů, viz kapitola 3 **Správa evidence výpočetního času**, je nastaveno spojení na relační databázi nového IS SCS. Tyto uzly zpracovávají účtovací soubory a získané informace vkládají do této databáze.

Původní myšlenka nasazení IS procesem *drop-in* nakonec byla nahrazena jiným postupem. Nový IS *SCS* byl nasazen do virtuální infrastruktury a spuštěn současně s původním informačním systémem. Data z *SCCS* byly replikovány jako pohledy do databáze systému *SCS*. Tím bylo umožněno provádění změn v obou systémech zároveň. Data však byla stále udržována v původní databázi IS. Služby jako *MOTD*, *it4ifree* či komunikace IS s portálem *Extranet* byly převáděny a testovány postupně.

Tímto způsobem byla umožněna možnost otestování všech vlastností nového IS za provozu, odladění případných chyb před jeho úplným nasazením a zároveň bylo možné využít dlouhodobě zaběhnutého původního IS.

Po postupném převedení všech služeb, několikadenním souběžném provozu obou systémů a migraci dat pomocí vytvořených migračních skriptů, mohl být původní IS odstaven z provozu.

3 Správa evidence výpočetního času

Jednou z úloh diplomové práce bylo vytvořit systém sběru dat pro evidenci výpočetního času napříč clustery superpočítačového centra. Pro tento účel byl vytvořen démon, zpracovávající výstupní soubory plánovače úloh, který vkládá výsledky do databáze systému SCS. K celému procesu je potřeba zmínit možnosti a podmínky získání výpočetní času neboli jádrohodin, jak tyto prostředky využívat a jaké omezení jsou stanovena v rámci IT4Innovations.

3.1 Přidělení výpočetních zdrojů

Výpočetní zdroje jsou přidělovány projektům v tzv. jádrohodinách neboli CH. O projekty si žádají uživatelé, kteří jsou označování jako hlavní řešitelé. Projektovou žádost je možno podat ve **veřejné grantové soutěži** nebo do fronty jejíž schválení probíhá na základě **rozhodnutí ředitelství**. Veřejná grantová soutěž je vypisována třikrát ročně pro zaměstnance vzdělávacích, výzkumných a vědeckých institucí. O přidělení času na základě rozhodnutí ředitelství lze zažádat kdykoliv. Jedná se o nepravidelné přidělování těchto zdrojů. Ucházet se může komerční i nekomerční sféra, a to v případě, že nelze využít veřejné grantové soutěže. V žádosti se uvádí alokace požadovaných jádrohodin jejíž plánované využití je doloženo vědeckou excelencí, výpočetní připraveností a společenskými dopady. Žádosti následně podléhají vědeckému, technickému a ekonomickému hodnocení určující úroveň jejich připravenosti. Z těchto hodnocení vychází rozhodnutí, zda bude žádostem výpočetní čas přidělen, či nikoliv. Toto zhodnocení provádí alokační komise. je také přihlíženo k žadatelům, kteří již v minulosti superpočítačové služby IT4Innovations využili, zejména k počtu předchozích žádostí, počtu registrovaných publikací navazujících na jejich projekty a k míře efektivity využití výpočetních zdrojů.

Pokud je projektová žádost schválena, stává se z ní projekt. Projekty mají přiděleny identifikátory, výpočetní čas, časové rozmezí, ve kterém projekt může využívat přidělený výpočetní čas a fronty, do kterých mohou být úlohy v rámci projektu zadávány.

3.2 Veličina výpočetních zdrojů

Přidělované výpočetní zdroje projektům jsou označovány jako jádrohodiny neboli CH a jsou základní jednotkou doby využití uzlů superpočítače.

Jedna jádrohodina je definována jako 1 alokované procesorové jádro po dobu 1 hodiny běhu. Celý alokovaný uzel (16 jader Anselm, 24 jader Salomon) po dobu 1 hodiny je roven 16 využitým jádrohodinám na superpočítači Anselm a 24 jádrohodinám na superpočítači Salomon.

Mezi úkoly vyvstal požadavek na faktorizaci využitých jádrohodin úloh během zpracovávání dat pro evidenci výpočetního času, jejíž detail je popsán v podkapitole 3.11.1. Faktorizací vznikly termíny normalizovaná jádrohodina NCH, což jsou spotřebované výpočetní zdroje po aplikaci faktoru a termín WCH, který faktorizaci opomíjí.

Vzorec výpočtu jádrohodin WCH můžeme vypočítat takto:

$$WCH = (RUW * RLN)/3600$$

RUW - Čas trvání běhu úlohy nebo rezervace v sekundách.
RLN - Počet alokovaných procesorových jader.

NCH jsou vypočteny pomocí normalizační faktoru, který je uveden v tabulce 1:

$$NCH = F * WCH$$

Tabulka 1: Faktorizace normalizovaných jádrohodin

Systém	F	Platnost
Salomon	1.00	2017-09-11 do 2018-06-01
Anselm	0.65	2017-09-11 do 2018-06-01

NCH byly zavedeny pro zacházení se systémy různého výkonu stejnou měrou. Je to účetní nástroj, který umožňuje zvýhodnění starších systémů. Všechny úlohy před 11.9.2017 mají faktor F nastaveny na hodnotu 1. V budoucnu se faktory F aktualizují, protože budou zprovozněny nové systémy. Očekává se, že faktor F bude postupem času pouze klesat. Pokud bude dále v této práci zmiňována jádrohodina, budeme vždy mluvit o normalizované jádrohodině.

3.3 Plánovač HPC úloh

V superpočítačovím centru IT4Innovations je využíváno softwaru PBS Professional, který plánuje, spouští a zpracovává pracovní úlohy uživatelů. PBS plánovač je rychlý, škálovatelný, bezpečný, odolný, podporující moderní infrastrukturu, middleware a aplikace [16]. Spravedlivým způsobem rozděluje dostupné výpočetní zdroje, s výhradou na omezení nastavením front, do nichž jsou úlohy zařazovány. Systém spravedlnosti zajišťuje, že uživatelé mohou týdně spotřebovat přibližně stejnou část výpočetních zdrojů.

3.4 Zadání úlohy do HPC systému

Pro zadání úlohy do fronty PBS plánovače se využívá příkaz *qsub*.

```
$ qsub -A PROJEKT_ID -q FRONTA -l select=64:ncpus=24,walltime=03:00:00 ./
    moje_uloha
```

Výpis 1: Zadání úlohy plánovači PBS

V příkladu zadání úlohy výše alokujeme 64 uzlů superpočítače s 24 jádry na uzel po dobu tří hodin. Úloha bude spuštěna ve specifikované frontě *FRONTA*. Spotřebované jádrohodiny budou připočteny k projektu *PROJEKT_ID*.

3.5 Fronty úloh

Fronty zajišťují prioritní a exkluzivní přístup k výpočetním zdrojům. Mají největší vliv na prioritu zařazení úlohy pro zpracování. Ostatní faktory úloh použité pro určení priority jejich spuštění, jako priorita spravedlivého sdílení nebo definování doby způsobilosti, nemohou konkurovat prioritě fronty.

Fronty superpočítačového centra IT4Innovations jsou rozděleny následovně [17]:

- **qexp** - Expresní fronta určena pro testování a spouštění velmi malých úloh. Není nutné udávat identifikátor projektu. Jedná se o tzv. bezprojektovou frontu. Pro využití této fronty není potřeba žádných zvláštních oprávnění. Maximální doba běhu úlohy v této frontě je 1 hodina.
- **qprod** - Fronta určená pro běžné produkční spuštění úloh. je požadováno zadání aktivního projektu s volnými výpočetními zdroji. Všechny uzly mohou být přístupné prostřednictvím této fronty. Má stanovenou středně velkou prioritu. Maximální doba běhu úlohy v této frontě je 48 hodin.
- **qlong** - Fronta určená pro dlouhodobé produkční úlohy. Vyžaduje zadání aktivního projektu s volnými výpočetními zdroji. Frontě je přiřazena středně velká priorita. Maximální doba běhu úlohy v této frontě je 144 hodin.
- **qmpp** - Fronta určená pro masivně paralelní úlohy. Projekt musí být aktivní s volnými výpočetními zdroji. Všechny uzly jsou přístupné pro tuto frontu. Této frontě je přidělena středně velká priorita. Maximální doba běhu úlohy je 4 hodiny.
- **qviz** - Vizualizační fronta určena pro zrychlené před/po zpracování pomocí grafických karet podporující OpenGL. Po přidělení přístupu k uzlu získá uživatel výhradní právo k jeho využití. Maximální doba využití vizualizačního uzlu je 2 hodiny.
- **qmic** - Fronta pro přístup k uzlům MIC. Projekt musí být aktivní s volnými výpočetními zdroji.
- **qnvidia** - Fronta určena pro přístup k akcelerovaným uzlům vybaveným NVIDIA grafickým procesorem. Projekt musí být aktivní s volnými výpočetními zdroji. Fronta má přiřazenou velmi vysokou prioritu. Úlohy jsou zařazeny k běhu před úlohami fronty *qexp*.
- **qfat** - Fronta určena pro přístup k uzlům s vyšší pamětí RAM.

- **qfree** - Fronta určena pro využití volných výpočetních zdrojů superpočítače po spotřebování všech alokovaných jádrohodin projektu. je požadováno zadání aktivního projektu. Spotřebované zdroje jsou projektu účtovány. Z této fronty lze přistupovat k výpočetním uzlům bez akcelerátorů. Frontě je přiřazena velmi nízká priorita. Maximální doba běhu úlohy je 12 hodin.

Výchozí čas běhu úloh je nastaven na polovinu maximální doby určené pro danou frontu. Pokud úloha překročí rezervovaný časový limit, je automaticky ukončena. Čas běhu úlohy může být změněn, pokud úloha čeká ve frontě před zpracováním, která je označena jako stav Q. Pokud je již úloha zpracovávána, není možné změnit její parametry.

Fronty mohou být nastaveny jako tzv. **bezprojektové**. Pro spuštění úlohy ve frontě s tímto příznakem není potřeba specifikovat identifikátor projektu a spotřebované jádrohodiny nejsou účtovány. Bezprojektová fronta dostupná běžným uživatelům superpočítačového centra je v současnosti nastavena pouze pro frontu **qexp**, jak můžeme vidět v tabulce 2. Fronty v této tabulce, které nebyly uvedeny ve výčtu výše slouží pro servisní úlohy a testy výpočetních uzlů superpočítače.

Tabulka 2: Přehled bezprojektových front

Název fronty	Bezprojektová fronta
ded	True
ded_mic	True
ded_test	True
qatlas	False
qexp	True
qfat	False
qfree	False
qlong	False
qmic	False
qmpp	False
qnvidia	False
qprace	False
qprod	False
qviz	False
SERVICE	False

Projektům jsou fronty přiřazovány ve dvou typech oprávnění a to:

- **S výpočetními zdroji** - Projekt s přidělenou frontou tohoto oprávnění může spouštět úlohy v dané frontě, pouze pokud má volné výpočetní zdroje k využití.
- **Bez výpočetních zdrojů** - ve frontě s tímto oprávněním může být spouštěna úloha i po vyčerpání všech alokovaných jádrohodin projektu.

3.6 Kontrola předání úlohy plánovači

Potvrzením příkazu *qsub* pro zadání úlohy PBS plánovači je **vyvolán hook** na metodu *check-access* nástroje SCS-API. Metoda provádí kontrolu, zda může být zadaná úloha předána plánovači PBS k dalšímu zpracování či nikoliv.

Příklad vyvolání metody check-access PBS plánovačem:

```
curl -i -H "Content-Type:application/json" -X POST \
  --data '{"pid":"DD-13-5","login":"test_user","queue":"qfat"}' \
  https://scs.it4i.cz/api/v1/check-access
```

Výpis 2: Volání metody check-access

Příklad návratové hodnoty metody check-access:

```
HTTP/1.1 200 OK
Content-Type: application/json

  "OK Access granted for regular queue."
```

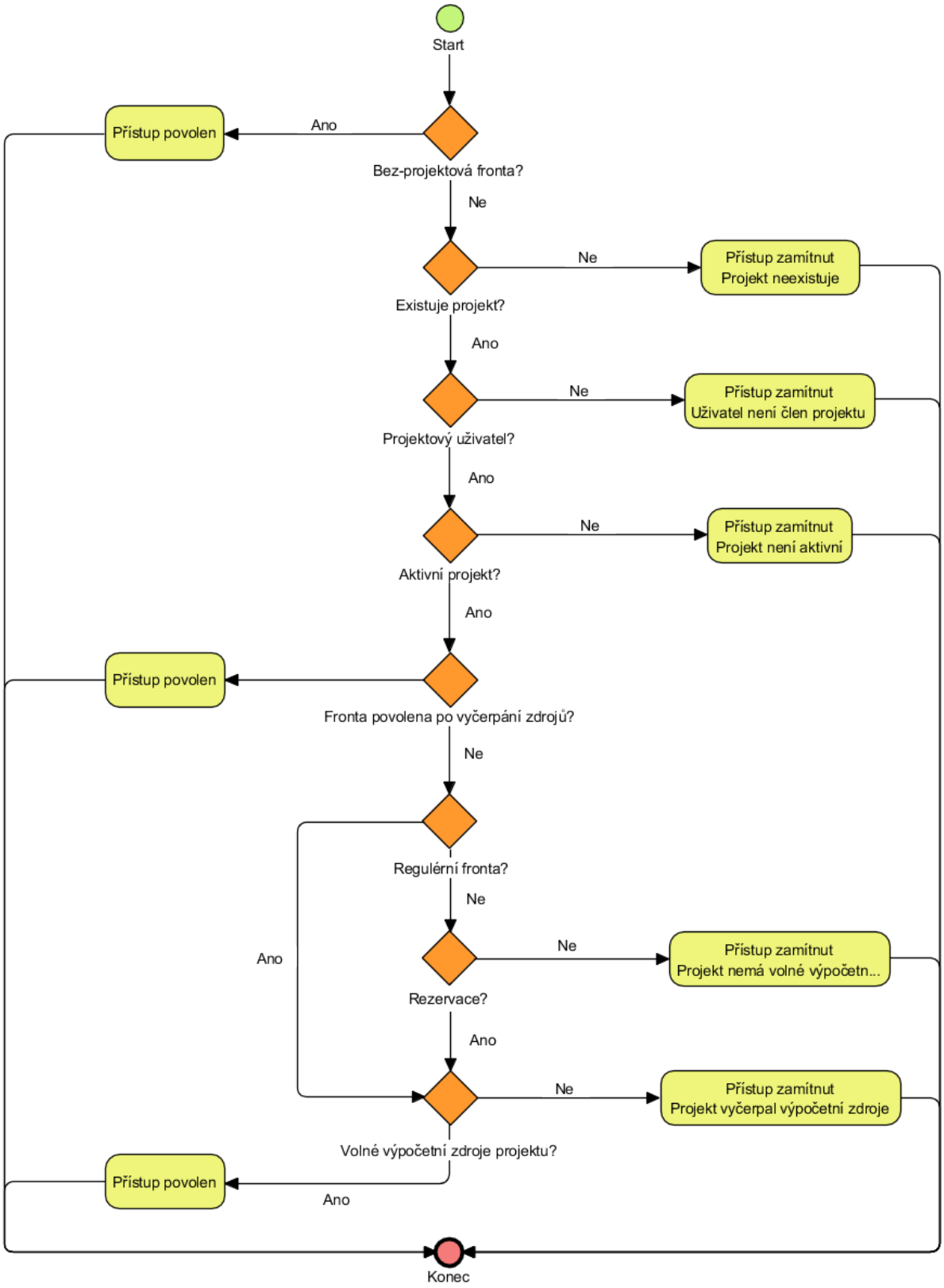
Výpis 3: Návratová hodnota metody check-access

Check-acces je metodou typu *POST*, příjimající tři parametry, dle tabulky 3. Těmito parametry je kontrolováno, zda definovaný projekt existuje, uživatel je členem tohoto projektu, projekt je aktivní s volnými výpočetními zdroji nebo má-li projekt oprávnění spouštět úlohy v zadané frontě. Pro jednotlivé části kontroly jsou v metodě vytvořeny SQL příkazy. Pokud příkaz z relační databáze SCS IS vrací záznamy, kontrola proběhla v pořádku. V opačném případě SQL příkaz navrací hodnotu *NULL*, což označuje nesplnění kontroly. Obrázek 4 zobrazuje vývojový diagram průběhu metody.

Tabulka 3: Parametry metody *check-access*

Název	Typ	Popis
login	string	uživatelké jméno
queue	string	název fronty
pid	string	identifikátor projektu

Pokud parametry úlohy splňují požadovaná kritéria, je úloha předána plánovači PBS. Úloha je nejprve vložena do fronty čekající na zpracování, kde jsou shromážděny všechny úlohy plánovače, dokud nedojde k jejich spuštění v HPC prostředí. Plánovač následně vybírá, kdy, kde a které úlohy mají být spuštěny. Úlohy jsou spouštěny dle nastavených pravidel, kdy PBS srovnává požadavky úloh s volnými dostupnými zdroji, prioritizuje jejich spuštění a přiřazuje dostupné zdroje podle určitých zásad. PBS také sleduje běh úlohy, který zapisuje do výstupních souborů.



Obrázek 4: Vývojový diagram metody *check-access*

3.7 Evidence výpočetního času

Zpracování a výpočet využitých jádrohodin projekty a uživateli superpočítačového centra IT4Innovations je prováděno na základě výstupních souborů PBS plánovače obsahující informace o běhu úloh. Tyto soubory jsou analyzovány a potřebná data jsou uložena do relační databáze SCS IS. Data jsou v této DB dále zpracovávána poskytovanými funkcemi databázového serveru MariaDB pro přehlednou evidenci a jednoduchou správu v IS.

3.8 Výstupní soubory plánovače PBS

Plánovač PBS poskytuje pro reportování běhu úlohy výstupní soubor, který můžeme v kontextu práce nad evidencí výpočetního času označovat také jako **účtovací soubor**, kde je každý záznam ukončen novým řádkem. Soubory jsou tvořeny pro jednotlivé dny a jsou ukládány do společné složky. Název výstupního souboru se odvíjí ode dne, kdy byl vytvořen a je uložen ve formátu *YYYYMMDD*, tedy například výstupní účtovací soubor ze dne 1.3.2018 nese název *20180301*. Záznamy v těchto souborech jsou uloženy v následujícím formátu:

datum-čas;typ-záznamu;id-úlohy;popis-úlohy

datum-čas
: Označuje datum a čas zapsání záznamu ve formátu *mm/dd/yyyy hh:mm:ss*.

typ-záznamu
: Příznak označující typ záznamu viz kapitola 3.9.

id-úlohy
: Identifikátor přiřazený úloze nebo rezervaci plánovačem, díky němuž jsme schopni rozlišit, na kterém superpočítači byl úkon proveden.

popis-úlohy
: Text popisující parametry úloh jsou odděleny mezerou ve formátu *klíč=hodnota*. Popis se odvíjí od typu záznamu. Není určeno žádné řazení těchto záznamů.

Příklad účtovacího souboru plánovače PBS Pro

```
08/16/2017 00:00:04;Q;1446601.isrv5;queue=qfree
08/16/2017 00:00:26;E;1446571.isrv5;user=test_user group=test_group account="DD
    -13-5" project=_pbs_project_default jobname=d-d-9 queue=qfree ctime
    =1502834060 qtime=1502834060 etime=1502834060 start=1502834098 exec_host=
    r33u04n868/0*24 exec_vnode=(r33u04n868:ncpus=24) Resource_List.fairshare=0
    Resource_List.mpiprocs=24 Resource_List.ncpus=24 Resource_List.nice=15
    Resource_List.nodect=1 Resource_List.place=free Resource_List.select=1:
    ncpus=24:mpiprocs=24:ompthreads=1 Resource_List.walltime=02:00:00 session
    =11146 end=1502834426 Exit_status=0 resources_used.cpupercent=2206
```

```
    resources_used.cput=02:06:24 resources_used.mem=947820kb resources_used.
    ncpus=24 resources_used.vmem=14174968kb resources_used.walltime=00:05:28
    eligible_time=00:00:00 run_count=1
08/16/2017 00:00:27;S;1446597.isrv5;user=test_user group=qianfan account="DD
    -13-5" project=_pbs_project_default jobname=d-d-c queue=qfree ctime
    =1502834372 qtime=1502834373 etime=1502834373 start=1502834427 exec_host=
    r36u34n967/0*24 exec_vnode=(r36u34n967:ncpus=24) Resource_List.fairshare
    =890334 Resource_List.mpiprocs=24 Resource_List.ncpus=24 Resource_List.nice
    =15 Resource_List.nodect=1 Resource_List.place=free Resource_List.select=1:
    ncpus=24:mpiprocs=24:ompthreads=1 Resource_List.walltime=02:00:00
    resource_assigned.ncpus=24
```

Výpis 4: Účtovací soubor plánovače PBS Pro

Ve výstupních souborech se mohou objevovat poškozené záznamy, které je během zpracování nutné ignorovat. Ty se obvykle vyskytují s chybně definovanou hlavičkou, což je jednoduché ošetřit. Další záznamy, které se během zpracování výstupních souborů ignorují, jsou ty, které neobsahují všechny povinné atributy v části záznamu označeného jako *popis-úlohy*.

3.9 Typy záznamů

Typ záznamu ve výstupním souboru reprezentuje stav průběhu úlohy. Podle typu záznamu je generován různý obsah popisující vlastnosti úloh *popis-úlohy*. Např. pro vytvořené rezervace budou v záznamu obsaženy jiné atributy, než je tomu pro úlohy. Těchto stavů je PBS plánovačem rozlišováno několik [18]:

A - Úloha přerušena serverem.

B - Zahájení běhu rezervace.

C - Vytvořen záchytný bod úlohy.

D - Úloha smazána.

E - Úloha ukončena.

F - Zdroje alokované rezervací byly uvolněny.

K - PBS vyžádal odstranění rezervace.

k - Uživatel vyžádal ukončení rezervace.

L - Licenční informace.

M - Přesunutí úlohy.

Q - Úloha zadána do fronty.

R - Úloha znovu spuštěna.

S - Spuštění úlohy zahájeno.

T - Úloha spuštěna ze záchytného bodu.

U - Vytvoření nepotvrzené rezervace na serveru.

Y - Rezervace potvrzena plánovačem.

Pouze úlohy vedeny v účtovacích souborech odpovídající stavům **B**, **E** a **R** jsou dále zpracovávány a účtovány.

3.10 Zpracování účtovacích souborů

Zpracování účtovacích souborů je prováděno **démonem it4i-acct**, který je napsán v interpretovaném programovacím jazyce Perl. Tento démon je uložen v projektu vedeného systémem správy verzí Git pod názvem *it4i-pbs-accouting*. V démonu je využíváno Perl extérních knihoven, označovaných jako moduly, zpřístupněné archivem CPAN.

Hlavním úkolem démonu je sledování a zpracování záznamů účtovacích souborů plánovače PBS, ze kterých jsou následně sestaveny SQL dotazy, které vkládají zpracované data do tabulek centrální databáze SCS IS, obsahující jednu databázovou tabulku pro úlohy (*accounting_pbs_jobs*) a druhou pro rezervace (*acccounting_pbs_reservations*), jak můžeme vidět na obrázku 5. Tabulky mají rozdílné atributy, jelikož se pro úlohy a rezervace liší atribut *popis-úlohy* ve výstupních účtovacích souborech.

Vždy, když je vložen nový záznam typu *B*, *E* nebo *R* do výstupního souboru PBS plánovače, se všechny spuštěné instance démonu it4i-acct pokusí spustit zpracovaný SQL dotaz pro vložení dat do centrální databáze. SQL dotaz je vytvořen s příznakem *IGNORE* a to za účelem zajištění dostupnosti účetního procesu na všech serverech PBS. Pokud tedy jeden ze serverů již vloží záznam do databáze a další server zpracuje tentýž záznam, je vložení dat druhým serverem ignorováno. Ignorování záznamu je rozpoznáno na základě jeho unikátního klíče, který se skládá z atributů identifikátoru, času zadání, počtu spuštění pro úlohy a identifikátoru s datem vytvoření pro rezervace.

Obrázek 5: Databázové tabulky pro úlohy a rezervace *check-access*

3.10.1 Spuštění démonu

Spuštěním démonu *it4i_acct* je vytvořen *thread pool*, ve kterém je nejprve vytvořeno vlákno pro zpracování nejnovějšího účtovacího souboru. Další vlákno je vytvořeno až pro nový účtovací soubor, tedy na začátku následujícího dne. V konfiguračním souboru je možné nastavit počet vláken, které si má démon uchovávat. Pokud démonu nastavíme zachování dvou vláken a máme dvě aktivní vlákna, jak je popsáno výše, tak po přechodu do dalšího dne, kdy je vytvořen další účtovací soubor, je vytvořeno třetí vlákno pro jeho zpracování. Nejstarší spuštěné vlákno je poté ukončeno. Toto chování můžeme vidět na následujícím výpisu:

```
2018/03/01 00:00:09 dm1 it4i_acct[14159]:325 Start process_one('PBS/accounting
    //20180301')
2018/03/01 00:00:10 dm1 it4i_acct[14159]:366 New thread started: 1

2018/03/02 00:00:08 dm1 it4i_acct[14159]:325 Start process_one('PBS/accounting
    //20180302')
2018/03/02 00:00:09 dm1 it4i_acct[14159]:366 New thread started: 2

2018/03/03 00:00:07 dm1 it4i_acct[14159]:325 Start process_one('PBS/accounting
    //20180303')
2018/03/03 00:00:08 dm1 it4i_acct[14159]:366 New thread started: 3
2018/03/03 00:00:10 dm1 it4i_acct[14159]:318 Received KILL signal. Exiting
    thread.
2018/03/03 00:00:11 dm1 it4i_acct[14159]:374 Killed thread: 1
```

Výpis 5: Vláknové zpracování účtovacích souborů

3.10.2 Zpracování účtovacího souboru vláknem

Vlákno zpracovávající účtovací soubor naváže spojení s centrální DB informačního systému SCS. Poté načte celý obsah zpracovávaného účtovacího souboru a dále sleduje pouze nově vkládané záznamy plánovačem PBS. Toto chování nám zajišťuje modul **File::Tail** pro jazyk Perl, který je schopen čtení z neustále aktualizovaného souboru.

Každý záznam z účtovacího souboru je předán analyzátoru, který buď navrací SQL dotaz pro vložení dat do databáze, nebo návratovou hodnotu, která definuje proč nebyl dotaz vytvořen. Korektní SQL dotaz je následně zaslán na DB skrze navázané spojení. Pokud zaslání dotazu na databázi selže (spojení se přeruší, vyprší nebo jiný problém), je celá akce opakována. V případě, že data do DB nemohou být vložena z důvodu chybějících povinných údajů, je tato událost zaznamenána do výstupního souboru démonu pro možnou analýzu.

3.10.3 Analyzátor záznamu

Analyzátoru je předán jeden konkrétní záznam z účtovacího souboru. Nejprve se kontroluje, jestli hlavička záznamu odpovídá správném formátu *datum-čas;typ-záznamu;id-úlohy;popis-úlohy*. Pokud je hlavička záznamu korektní, načtou se společné atributy záznamu pro úlohy a rezervace. Poté jsou dle typu záznamu doplněny zbývající atributy specifické pro úlohu či rezervaci. Pokud některý z atributů v *popis-úlohy* chybí, je jeho hodnota nastavena na *NULL*. Nakonec analyzátor vytvoří SQL dotaz a navrátí jej. V případě, že analyzátor rozpozná, že se nejedná o úlohu nebo rezervaci, je akce analyzátoru ukončena s návratovou hodnotou *NULL*.

3.10.4 Shrnutí zpracování výstupních souborů démonem

Démon pro zpracování účtovacích souborů je v současnosti provozován na čtyřech serverech. Dva servery [*dm1, dm2*] jsou určeny pro zpracování účtovacích souborů superpočítače Anselm a další dva [*isrv5, isrv6*] pro superpočítač Salomon. Na všech těchto serverech jsou vytvářeny a shromažďovány výstupní soubory PBS plánovače.

Změny v démonu a jejich distribuce na zmíněné servery jsou řízeny nástroji *Ansible* a *Puppet*, které provádějí víceuzlové nasazení softwaru a správu konfiguračních souborů. Pokud např. potřebujeme upravit skript pro zpracování dalšího typu záznamu, pouze upravíme aplikované předpisy v těchto nástrojích, což je velice jednoduché díky verzovacímu systému Git a změny se automaticky distribuují na servery. Tímto je automatizována celková správa démonu a především jeho řízení změn.

Démon obsahuje konfigurační soubor, jímž jsme schopni nastavit např. úroveň výpisu informací výstupních souborů (*TRACE, DEBUG, INFO, WARN, ERROR, FATAL*), které nám ulehčují ladění programu v různých fázích vývoje a omezení výstupu během jeho standardního běhu. Dále můžeme definovat cestu pro vyhledávání účtovacích souborů, cestu pro uložení samotného výstupního souboru démonu, počet aktivních vláken, řetězec spojení s DB serverem IS, vyhledávané atributy pro úlohy či rezervace v záznamech a názvy databázových tabulek, do kterých jsou zpracovávaná data vkládána.

Způsob, kterým byl popsán průběh zpracování účtovacích souborů až po vložení záznamu do centrální DB, kde jsou data ještě dále zpracovávána, nám zajišťuje aktuální, rychlé a přehledné vytížení superpočítačového centra. V informačním systému SCS můžeme analyzovat jednotlivé úlohy uživatelů a kontrolovat dostupné prostředky projektů v reálném čase, což je velice důležité především pro kontrolu zadávání úloh plánovači.

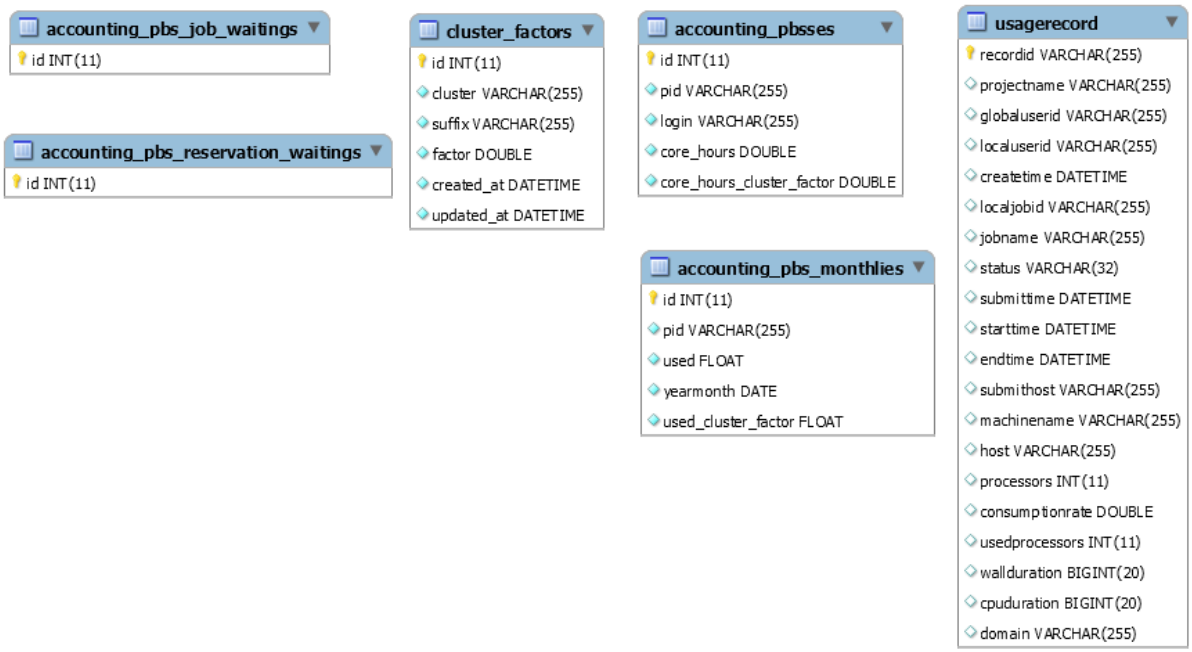
3.11 Zpracování záznamů databázovým serverem

Po té co je záznam z účtovacího souboru zpracován démonem *it4i_acct* a vložen do centrální DB, prochází záznam ještě několika fázemi úprav, které jsou prováděny uloženými programy na databázovém serveru. Pro zpracování je využito dvou typů uložených programů a to:

- **Spouštěč**
  Spouštěč neboli *trigger* je databázová funkce, která je svázána s tabulkou. Úloha spouštěče je provedena během aktivování definované akce nad svázanou tabulkou, jako je vložení nebo aktualizace záznamu.

- **Událost**
  Událost je úloha, kterou server spustí podle stanoveného časového plánu.

Relační schéma DB je rozšířeno o další tabulky znázorněné na obrázku 6, spolu s popisem v tabulce 4, které nám ulehčují správu agregovaných dat v IS.



Obrázek 6: DB schéma pro agregaci výpočetního času

Tabulka 4: Popis DB schématu pro agregaci výpočetního času

Název tabulky	**Popis**
accounting_pbs_jobs_waitings	Fronta nově vložených úloh čekajících na další zpracování
accounting_pbs_reservation_waitings	Fronta nově vložených rezervací čekajících na další zpracování
cluster_factors	Přehled aktuálních účtovacích faktorů pro superpočítače
accounting_pbsses	Souhrn využitých jádrohodin uživatele k projektu
accounting_pbs_monthlies	Evidence měsíčního využití jádrohodin projekty
usagerecord	Evidence využití jádrohodin PRACE projekty

V následujících podkapitolách budou popsány jednotlivé akce probíhající na databázovém serveru, jako je naplnění tabulek agregovanými daty, jednodušší a přehlednější evidence využitých zdrojů superpočítačového centra.

3.11.1 Nastavení faktoru záznamu

Záznam z účtovacího démonu vkládaný do centrální databáze IS, vyvolává před samotným uložením databázový spouštěč vykonávající nastavení faktoru pro úlohy a rezervace. Faktor záznamu je nastaven podle dat v databázové tabulce *cluster_factors*, jejíž obsah je zobrazen v tabulce 5. Spolu s nastaveným faktorem je celý záznam uložen do DB tabulky pro úlohy nebo rezervace.

Tabulka 5: Tabulka faktorů výpočetního času superpočítačů Anselm a Salomon

Cluster	Suffix	Factor
Anselm	.dm2$	0.65
Anselm	.login1$	0.65
Salomon	.isrv5$	1
Salomon	.isrv5.head.smc.salomon$	1
Unknown	.srv11$	1
Unknown	.srv11.bullx$	1

Spouštěč vyhledá příslušný faktor díky regulárnímu výrazu uvedeným v atributu **suffix** nad identifikátorem úlohy nebo rezervace. Pokud by identifikátor neodpovídal žádnému ze záznamů, je faktor nastaven na implicitní hodnotu 1. Suffix nám jednoznačně určí, na kterém ze superpočítačů byla úloha či rezervace vytvořena. V současné době se porovnávají primárně dva typy regulárního výrazu a to *.dm2$* pro záznamy superpočítače Anselm a *.isrv5$* pro záznamy

superpočítače Salomon. Ostatní záznamy tabulky 5 jsou zde vedeny z historických důvodů, kdy se testovalo nasazení superpočítačů a konfigurace plánovače PBS pro ně.

3.11.2 Fronta záznamů

Po té co je záznam i s faktorem vložen do tabulek úloh nebo rezervací je celočíselný ID odkaz na záznam, tedy primární klíč, uložen do fronty čekající na zpracování, která je reprezentována tabulkami s příznakem *__waitings*, jak je viditelné v tabulce 4. Fronty jsou opět rozděleny na tabulky pro úlohy a rezervace.

Fronty jsou dále zpracovávány a výsledky uloženy do agregační tabulky *accounting__pbsses* identifikující uživatele, projekt nad kterým spouštěl úlohy a počet celkově spotřebovaných jádrohodin. Akce plnící tabulku front je vyvolána spouštěčem, který je aktivován po vložení záznamu do tabulek pro úlohy a rezervace, provádějící dodatečné kontroly před samotným vložením záznamu do fronty.

Naplnění fronty úloh

Pokud definovaná fronta úlohy není bezprojektová, nebo se nejedná o rezervační frontu, je ID záznamu uloženo do fronty úloh pro další zpracování.

Bezprojektovým frontám nejsou využité jádrohodiny úloh účtovány. Jedná se o využívaní výpočetních zdrojů „zdarma“. Zadáním úlohy pro zpracování v rámci rezervace, se také jádrohodiny k projektu nezapočítávají. Vytvořením rezervace jsou projektu automaticky přiřazeny jádrohodiny dle vzorce v kapitole 3.2.

Jestli projekt, pod kterým je úloha spuštěna, spadá pod organizaci *PRACE*, je úloha dále evidována do tabulky *usagerecord*, která je pevně definovaná napříč všemi metacentry, kde organizace PRACE participuje.

Naplnění fronty rezervací

Tento spouštěč provádí kontrolu, jestli definovaný projekt může rezervaci využívat. Pokud ano, je ID rezervace vloženo do fronty rezervací pro další zpracování.

3.11.3 Zpracování front záznamů

Fronty úloh a rezervací čekající na další zpracování jsou obslouženy událostmi databázového serveru. Tyto události nesou stejný název jako tabulky front s přidanou příponou *__process*. Úloha událostí je spuštěna podle nastaveného časového plánu každou sekundu.

Funkcionalita událostí pro zpracování úloh a rezervací

Vlákno uzamkne tabulku fronty pomocí sdíleného zámku, zkopíruje všechny záznamy z fronty do samostatné dočasné tabulky, odstraní původní záznamy a odemkne sdílený zámek. Jinými slovy se záznamy front přesouvají do dočasné tabulky a fronty mohou být dále plněny. Pokud během aktivního sdíleného zámku přistoupí další událost k této tabulce, je zpracování nové

události ukončeno, místo toho, aby očekával uvolnění sdíleného zámku. V praxi je sdílený zámek velmi rychle uvolněn, jelikož ID záznamy úloh/rezervací se ihned přesunou. Poté může každý podproces nezávisle zpracovávat obrovské množství nových identifikačních čísel záznamů. Zpracování by mohlo trvat jakkoliv dlouho, jelikož neblokuje obsluhu zpracování novými událostmi. Důsledek tohoto chování může vytvořit mnoho paralelních vláken, které zpracovávají různé ID úloh/rezervací a kalkulují agregované hodnoty.

Po té, co událost naplní dočasnou tabulky ID hodnotami pro úlohy/rezervace, je pro každý tento záznam vypočítáno, kolik spotřeboval WCH a NCH. Vypočtení využitých jádrohodin odpovídá vzorcům uvedeným v kapitole 3.2 **Veličina výpočetních zdrojů**.

Využité jádrohodiny jsou přiřazeny k *loginu* uživatele a identifikátoru projektu, nad kterým byla úloha/rezervace vytvořena. Tento výsledný záznam je uložen do tabulky *accounting_pbsses*, která tyto údaje shromažďuje. Z této tabulky je tedy velmi jednoduché zjistit, kolik jádrohodin bylo uživatelem nad definovaným projektem využito. Příklad si můžeme ukázat na následující tabulce 6.

Tabulka 6: Souhrn využitého výpočetního času projekty

Project	Login	WCH	NCH
DD-13-5	test-user-1	2880.00	2880.00
DD-13-5	test-user-2	768.00	768.00
DD-13-5	test-user-3	6480.00	6180.53
DD-12-1	test-user-1	25200.00	25200.00
OPEN-3-2	test-user-4	3.62	2.71
SERVICE	test-user-3	107.92	70.15

Celé zpracování události je zahrnuto v transakci. Pokud by během vykonání nastal jakýkoliv problém, jsou identifikátory záznamu zpět navráceny do původní tabulky pro fronty.

Po vykonání události jsou všechny akce spojené se záznamem úlohy nebo rezervace ukončeny.

3.12 Měsíční souhrny využitého výpočetního času

Součástí návrhu evidence výpočetního času superpočítačového centra bylo vypracování přehledných měsíčních souhrnů využitých jádrohodin projekty. K tomu byla vytvořena tabulka *accounting_pbs_monthlies*, která je přepočítávána ve 20 minutových intervalech. Agregovaná data pro tuto tabulku jsou získávány z tabulky uchovávající všechny úlohy, která v současnosti obsahuje přibližně 8 miliónů záznamů. Logika naplnění tabulky měsíčních souhrnů je rozdělena do dvou fází:

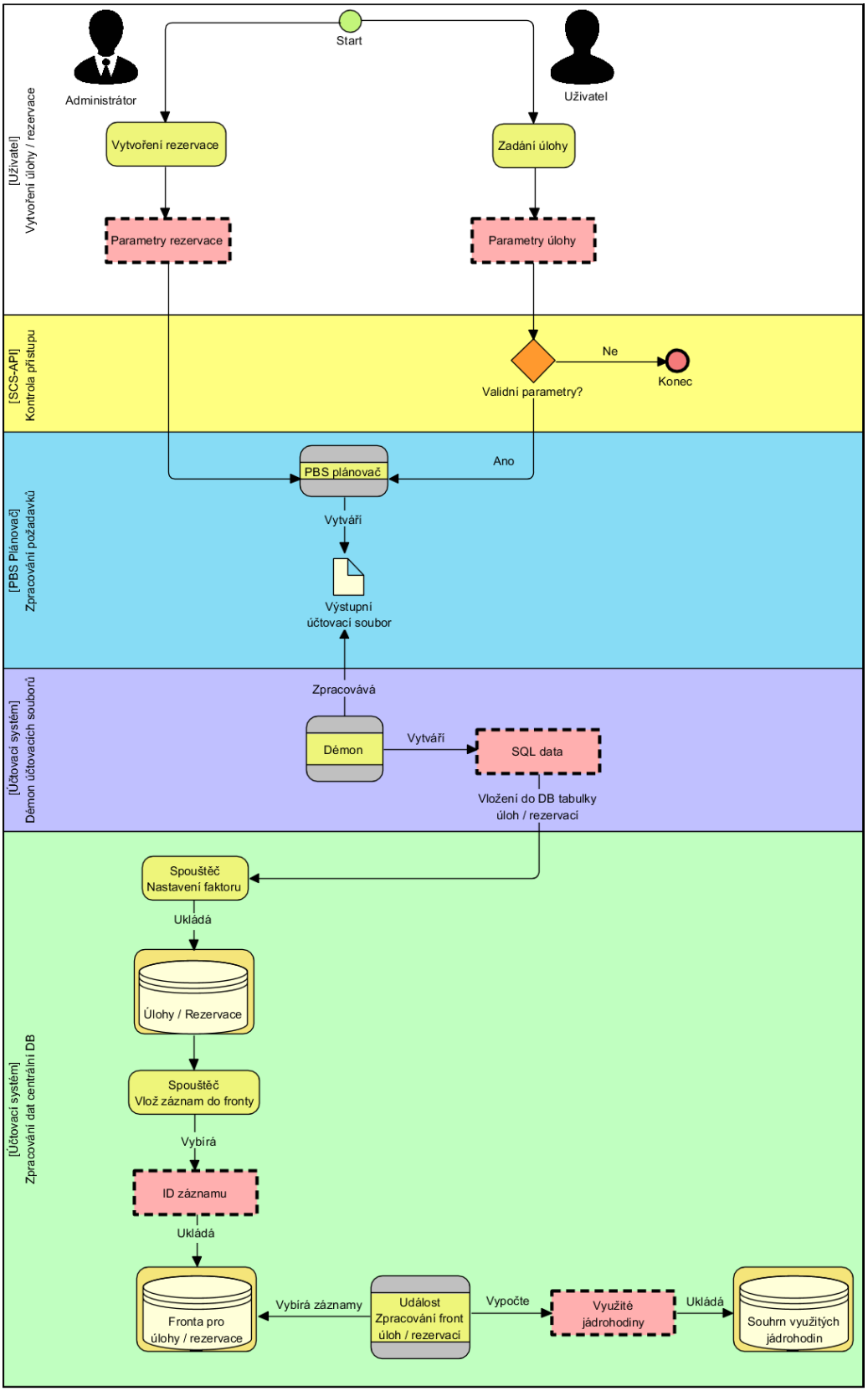
1. **Naplnění tabulky měsíci**
   Pro každý projekt je vytvořena kolekce dat ve formátu *YYYY-MM-01*, kdy první datum indikuje začátek platnosti projektu a koncové datum je vygenerováno na datum ukončení platnosti projektu plus jeden měsíc navíc, kdy mohou zadané úlohy projektu ještě doběhnout.

2. **Naplnění tabulky využitými jádrohodinami**
   V tomto kroku vypočítáváme měsíčně využité jádrohodiny projektů. Jsou iterovány všechny úlohy, které byly spuštěny v regulérních frontách, shlukovány podle identifikátoru projektu a data zadání úlohy ve zmiňovaném formátu. Celý tento výběr je vkládán do tabulky pro tyto souhrny. Pokud během vkládání záznamů již v tabulce existuje záznam se stejným unikátním klíčem, který je definován jako *identifikátor projektu* a *datum YYYY-MM-01*, je pro něj aktualizována hodnota využitých jádrohodin. K situaci, kdy při vkládání narazíme na duplicitní klíč, dochází téměř ve všech případech, jelikož jsme si tabulky s odpovídajícími daty naplnili již v prvním kroku tohoto úkonu. Jelikož tato operace pracuje nad velkým množstvím úloh a pro každou z těchto úloh musíme kontrolovat, jestli byla spuštěna ve správné frontě, je trvání této akce zdlouhavé a dosahuje času zpracování přibližně 20 sekund. V budoucnu bude potřeba tuto problematiku podchytit, stejně jako rapidně rostoucí velikost tabulky obsahující informace o spuštěných úlohách.

3.13 Souhrn správy evidence výpočetního času

V této kapitole jsme si popsali průběh od zadání úlohy superpočítači, která má být spuštěna, až po její evidenci v centrálním databázovém systému. Celkový průběh je zachycen obecným zjednodušeným diagramem na obrázku 7.

Jednotlivé moduly pracující s plánovačem PBS či jeho výstupy jsou koncipovány způsobem, že pokud dojde k jejich výpadku, tak neovlivní funkci samotného plánovače. Moduly jsou dále navrženy způsobem, aby se po jejich zotavení daly všechny záznamy, které nebyly během výpadku zpracovány, zpětně vyhodnotit a uložit do centrální databáze SCS IS. Na druhou stranu výpadek API, které kontroluje zadávání úloh, způsobí, že všechny zadané úlohy i s nereálnými parametry jsou předány plánovači ke zpracování. Tím může dojít k nekonzistentním či fiktivním datům ve výstupních souborech plánovače.



Obrázek 7: Zjednodušený průběh zpracování úlohy/rezervace účtovacím systémem

4 Závěr

Vývoj nového informačního systému, nahrazujícího původní verzi, mohu označit za úspěšný jak z pohledu získání nových zkušeností, tak i z pohledu zjednodušení správy superpočítačového centra IT4Innovations. SCS IS je spuštěn v produkčním provozu a denně jej pro svou práci využívá administrátorský tým. Také API server je v neustálém stabilním provozu a jsou na něj zasílány požadavky informačním systémem, uživateli centra IT4Innovations, PBS plánovačem úloh a službami pro správu plánovaných výpadků a MOTD. Podchycením automatizace nasazení vývojového prostředí, řízení vývoje systémem správy verzí, nastavením kontinuální integrace a vývojem IS stojícím na architektuře MVC, je snadné rozšířit tým vývojářů a rychle je uvést do struktury tohoto komplexního systému.

Díky procesům zpracovávajícím evidenci výpočetního času, která je vedena v informačním systému SCS, můžeme analyzovat úlohy uživatelů a sledovat využité výpočetní zdroje projektů téměř v reálném čase. Dále byl zaveden termín normalizovaných jádrohodin NCH, jehož cílem je zvýhodnění využívání výpočetních prostředků na starších systémech. Aktuálně je sleva pro výpočetní zdroje udělena pro superpočítač Anselm.

V současné době se vývoj SCS IS nachází ve stavu, kdy se pracuje na převodu řízení a správy uživatelů. Uživatelé jsou nyní uloženi v databázi LDAP. Po převodu by měla být celá správa uživatelů vedena za pomocí IS. S tím souvisí další plán vývoje, a to zpřístupnění SCS IS běžným uživatelům, kteří budou moci využívat jeho služeb jako žádost o navýšení zdrojů, prodloužení projektů nebo přidávání uživatelů k projektům. Došlo by tak k nahrazení stávajícího portálu Extranet, který také není možno nadále rozvíjet a v současné době slouží k těmto účelům.

Velikost databázové tabulky pro evidenci úloh spuštěných v superpočítačovém centru narůstá rychlým tempem. Během zpracování diplomové práce vzrostl počet evidovaných úloh přibližně o jednonásobek oproti celkovému běhu superpočítačového centra, které započalo v roce 2013. Bude nutné navrhnout způsob škálovatelnosti podchycující tento nárůst dat, aby bylo možné v IS nadále provádět rychlý výběr dat a operace nad nimi.

Literatura

[1] KIT Blake. *Introduction - Zope Scalability [online]*. c2007. [cit. 2017-11-15]. <http://infrae.com/presentations/present_Gebruikersdag_Adam>

[2] SPETTOLI, Giacomo. *Introduction - Plone Documentation [online]*. c2014. [cit. 2018-02-27]. <https://docs.plone.org/5/en/intro/index.html#what-is-plone>

[3] QUINTAGROUP.*Plone Zope - Quintagroup [online]*. c2004. [cit. 2018-01-20]. <http://quintagroup.com/cms/plone/zope>

[4] The SiteGround People. *What is MySQL Tutorial [online]*. c2008. [cit. 2018-04-22]. <https://www.siteground.com/tutorials/php-mysql/mysql/>

[5] CHEASLEY, Adam. *ZODB Database-Plone Documentation [online]*. c2014. [cit. 2017-09-13]. https://docs.plone.org/develop/plone/persistency/database.html

[6] Zope Community. *ZSQL Users guide [online]*. c2002. [cit. 2018-02-25]. <http://old.zope.org/ProductFiles/Documentation/Guides/ZSQL>

[7] PREISLER, Korbinian. *Plone Documentation [online]*. c2014. [cit. 2018-03-01]. <https://docs.plone.org/manage/deploying/zope.html>

[8] ŠTRÁFELDA, Jan. *Co je apache server [online]*. c2005. [cit. 2018-01-06]. <http://www.adaptic.cz/znalosti/slovnicek/apache-server>

[9] Codeacademy. *Request-Response Cycle [online]*. c2016. [cit. 2017-12-11]. <https://www.codecademy.com/articles/request-response-cycle-dynamic>

[10] BEAM, Andrew *RubyOnRails.cz [online]*. c2014. [cit. 2018-02-27]. <http://rubyonrails.cz/>

[11] CHULKI Lee. *Ruby/Rack web server built for concurency [online]*. c2013. [cit. 2018-02-27]. <https://github.com/puma/puma>

[12] MariaDB Foundation. *About MariaDB [online]*. c2015. [cit. 2017-10-08]. <https://mariadb.org/about/>

[13] NGINX LEADERSHIP. *Welcome to NGINX Wiki! [online]*. c2017. [cit. 2018-03-22]. <https://www.nginx.com/resources/wiki/>

[14] RSpec. *RSpec Documentation [online]*. c2015. [cit. 2018-04-19] <http://rspec.info/documentation/>

[15] VALIM, José. *Documentation for devise [online]*. c2009. [cit. 2017-09-14]. <http://www.rubydoc.info/github/plataformatec/devise>

[16] PBS Proffesional. *PBS Professional Open Source Project [online]*. c2017. [cit. 2018-04-02]. <http://www.pbspro.org/>

[17] IT4Innovations. *Resources Allocation Policy - IT4Innovations Documentation [online]*. c2017. [cit. 2017-09-07]. <https://docs.it4i.cz/salomon/resources-allocation-policy>

[18] Altair Engineering INC. *PBS Professional Reference Guide [online]*. c2003. [cit. 2018-03-04]. <https://pbsworks.com/pdfs/PBSReferenceGuide13.0.pdf>

Přílohy

Přílohou této diplomové práce je CD obsahující následující součásti:

- Zdrojové kódy aplikací ve složkách *scs-ror* obsahující aplikaci SCS IS, *scs-api* obsahující API server a *it4i-pbs-accounting* obsahující démona účtovacích souborů.
- SQL soubor obsahující databázové schéma SCS IS *structure.sql*.
- Ukázky nového IS ve složce *ukazky*.